SUSE Linux Enterprise Server 10: Network Services

Novell Training Services

COURSE 3074

www.novell.com

AUTHORIZED COURSEWARE

100-005064-001
Version 1

Novell®

Proprietary Statement

Novell, Inc.
1800 South Novell Place
Provo, UT 84606-2399

Disclaimer

Novell, Inc. makes no representations or warranties with respect to the contents or use of this manual, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose.

Further, Novell, Inc. reserves the right to revise this publication and to make changes in its content at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any NetWare software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose.

Further, Novell, Inc. reserves the right to make changes to any and all parts of NetWare software at any time, without obligation to notify any person or entity of such changes.

This Novell Training Manual is published solely to instruct students in the use of Novell networking software. Although third-party application software packages are used in Novell training courses, this is for demonstration purposes only and shall not constitute an endorsement of any of these software applications.

Further, Novell, Inc. does not represent itself as having any particular expertise in these application software packages and any use by students of the same shall be done at the students' own risk.

Software Piracy

Throughout the world, unauthorized duplication of software is subject to both criminal and civil penalties.

If you know of illegal copying of software, contact your local Software Antipiracy Hotline.

For the Hotline number for your area, access Novell's World Wide Web page at http://www.novell.com and look for the piracy page under "Programs."

Or, contact Novell's anti-piracy headquarters in the U.S. at 800-PIRATES (747-2837) or 801-861-7101.

Trademarks

Novell, Inc. has attempted to supply trademark information about company names, products, and services mentioned in this manual. The following list of trademarks was derived from various sources.

Novell, Inc. Trademarks

Novell, the Novell logo, NetWare, BorderManager, ConsoleOne, DirXML, GroupWise, iChain, ManageWise, NDPS, NDS, NetMail, Novell Directory Services, Novell iFolder, Novell SecretStore, Ximian, Ximian Evolution and ZENworks are registered trademarks; CDE, Certified Directory Engineer and CNE are registered service marks; eDirectory, Evolution, exteNd, exteNd Composer, exteNd Directory, exteNd Workbench, Mono, NIMS, NLM, NMAS, Novell Certificate Server, Novell Client, Novell Cluster Services, Novell Distributed Print Services, Novell Internet Messaging System, Novell Storage Services, Nsure, Nsure Resources, Nterprise, Nterprise Branch Office, Red Carpet and Red Carpet Enterprise are trademarks; and Certified Novell Administrator, CNA, Certified Novell Engineer, Certified Novell Instructor, CNI, Master CNE, Master CNI, MCNE, MCNI, Novell Education Academic Partner, NEAP, Ngage, Novell Online Training Provider, NOTP and Novell Technical Services are service marks of Novell, Inc. in the United States and other countries. SUSE is a registered trademark of SUSE Linux AG, a Novell company. For more information on Novell trademarks, please visit http://www.novell.com/company/legal/trademarks/tmlist.html.

Other Trademarks

Adaptec is a registered trademark of Adaptec, Inc. AMD is a trademark of Advanced Micro Devices. AppleShare and AppleTalk are registered trademarks of Apple Computer, Inc. ARCserv is a registered trademark of Cheyenne Software, Inc. Btrieve is a registered trademark of Pervasive Software, Inc. EtherTalk is a registered trademark of Apple Computer, Inc. Java is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries. Linux is a registered trademark of Linus Torvalds. LocalTalk is a registered trademark of Apple Computer, Inc. Lotus Notes is a registered trademark of Lotus Development Corporation. Macintosh is a registered trademark of Apple Computer, Inc. Netscape Communicator is a trademark of Netscape Communications Corporation. Netscape Navigator is a registered trademark of Netscape Communications Corporation. Pentium is a registered trademark of Intel Corporation. Solaris is a registered trademark of Sun Microsystems, Inc. The Norton AntiVirus is a trademark of Symantec Corporation. TokenTalk is a registered trademark of Apple Computer, Inc. Tru64 is a trademark of Digital Equipment Corp. UnitedLinux is a registered trademark of UnitedLinux. UNIX is a registered trademark of the Open Group. WebSphere is a trademark of International Business Machines Corporation. Windows and Windows NT are registered trademarks of Microsoft Corporation.

All other third-party trademarks are the property of their respective owners.

Contents

SECTION 5 Use OpenSLP

SECTION 6 Monitor Network Traffic

Introduction

In *SUSE Linux Enterprise Server 10: Networking Services* (Course 3074), you learn advanced Linux skills focused on network services you need to administer SUSE Linux Enterprise Server 10.

These skills, along with those taught in *SUSE Linux Enterprise Server 10: Security* (Course 3075), prepare you to take the Novell Certified Linux Engineer 10 (Novell CLE 10) certification practicum test.

For more information on CLE 10, visit

http://www.novell.com/training/certinfo/cle10

The contents of your student kit include the following:

- *SUSE Linux Enterprise Server 10: Networking Services* Manual
- *SUSE Linux Enterprise Server 10: Networking Services* Workbook
- *SUSE Linux Enterprise Server 10: Networking Services* Course DVD
- *SUSE Linux Enterprise Server 10* Product DVD
- *SUSE Linux Enterprise Desktop 10* Product DVD

The *SUSE Linux Enterprise Server 10 Networking Services* Course DVD contains a VMware Workstation SUSE Linux Enterprise Server 10 server (in the directory **vmware**) that you can use with the *SUSE Linux Enterprise Server 10: Networking Services* Workbook outside the classroom to practice the skills you need to take the Novell CLE 10 Practicum.

Instructions for setting up a self-study environment are in the **setup** directory on the Course DVD.

Course Objectives

This course teaches you how to perform the following SUSE Linux network services tasks for SUSE Linux Enterprise Server 10:

1. Configure a DNS Server Using BIND.
2. Use DHCP to Manage Networks.
3. Manage OpenLDAP.
4. Configure a Mail Server.
5. Use OpenSLP.
6. Monitor Network Traffic.

These are common tasks performed by an experienced SUSE Linux Enterprise Server administrator responsible for network services in an enterprise environment. Half of the final day of class is reserved for a "LiveFire" exercise that provides a set of scenarios to test your SUSE Linux Enterprise Server 10 administration skills and prepare you to take the Novell CLE 10 Practicum.

Audience

The primary audience for this course are Novell Certified Linux Professionals (CLPs).

For more information on CLP, visit

http://www.novell.com/training/certinfo/clp

Certification and Prerequisites

This course helps you prepare for the Novell Certified Linux Engineer 10 (CLE 10) Practicum exam, called the *Practicum*. The Novell CLE 10 is a higher-level certification for people who passed the Novell CLP Practicum.

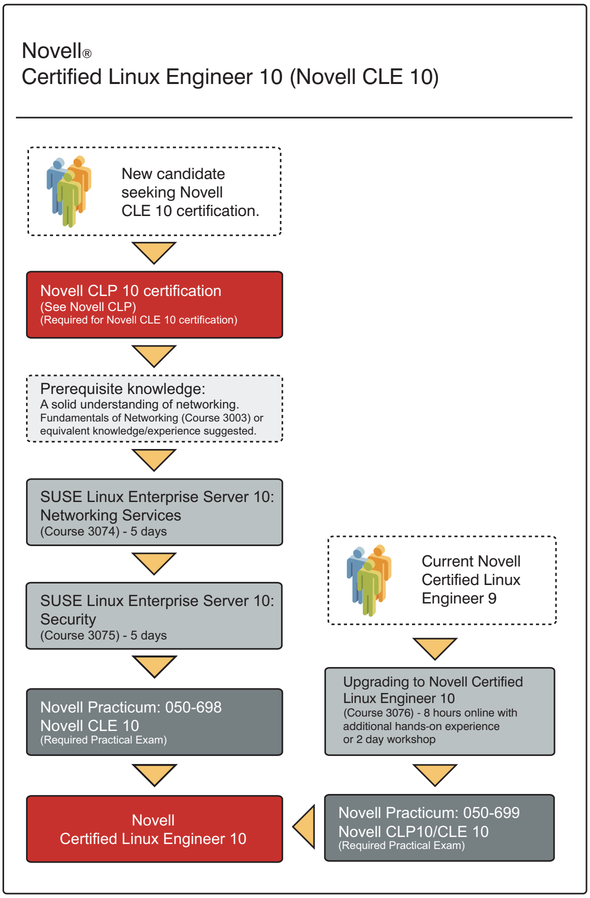
As with all Novell certifications, course work is recommended. To achieve the certification, you are required to pass the Novell CLE 10 Practicum (050–698).

The Novell CLE 10 Practicum is a hands-on, scenario-based exam where you apply the knowledge you have learned to solve real-life problems—demonstrating that you know what to do and how to do it.

The practicum tests you on objectives of this course (*SUSE Linux Enterprise Server 10: Networking Services* – Course 3074) and those covered in *SUSE Linux Enterprise Server 10: Security* (Course 3075).

The following illustrates the training and testing path for Novell CLE:

Figure Intro-1



The exercises in this course assume that students are familiar with text editors. They do not list every single step but assume that the students are already familiar with Linux system administration and does not need a detailed description of basic steps.

Before attending this course, you should have attended the following courses:

- *SUSE Linux Enterprise Server 10 Fundamentals* (Course 3071)
- *SUSE Linux Enterprise Server 10 Administration* (Course 3072)
- *SUSE Linux Enterprise Server 10 Advanced Administration* (Course 3073)

Having passed the Novell CLP is also acceptable preparation.

SUSE Linux Enterprise Server 10: Networking Services (Course 3074) is the first course of the CLE10 track. The final course is *SUSE Linux Enterprise Server 10: Security* (Course 3075).

For more information about Novell certification programs and taking the Novell CLP 10 and CLE 10 Practicum exam, see http://www.novell.com/training/certinfo/.

SUSE Linux Enterprise Server 10 Support and Maintenance

The copy of SUSE Linux Enterprise Server 10 you receive in your student kit is a fully functioning copy of the SUSE Linux Enterprise Server 10 product.

However, to receive official support and maintenance updates, you need to do one of the following:

- Register for a free registration/serial code that provides you with 30 days of support and maintenance.
- Purchase a copy of SUSE Linux Enterprise Server 10 from Novell (or an authorized dealer).

You can obtain your free 30-day support and maintenance code at http://www.novell.com/products/linuxenterpriseserver/eval.html.

You will need to have or create a Novell login account to access the 30-day evaluation.

Novell Customer Center

Novell Customer Center is an intuitive, web-based interface that helps you to manage your business and technical interactions with Novell. Novell Customer Center consolidates access to information, tools and services such as:

- Automated registration for new SUSE Linux Enterprise products
- Patches and updates for all shipping Linux products from Novell
- Order history for all Novell products, subscriptions and services
- Entitlement visibility for new SUSE Linux Enterprise products
- Linux subscription-renewal status
- Subscription renewals via partners or Novell

For example, a company might have an administrator who needs to download SUSE Linux Enterprise software updates, a purchaser who wants to review the order history and an IT manager who has to reconcile licensing. With Novell Customer Center, the company can meet all these needs in one location and can give each user access rights appropriate to their roles.

You can access the Novell Customer Center at http://www.novell.com/center.

SUSE Linux Enterprise Server 10 Online Resources

Novell provides a variety of online resources to help you configure and implement SUSE Linux Enterprise Server 10.

These include the following:

- http://www.novell.com/products/linuxenterpriseserver/

  This is the Novell home page for SUSE Linux Enterprise Server 10.

- http://www.novell.com/documentation/sles10/index.html

  This is the Novell Documentation web site for SUSE Linux Enterprise Server 10.

- http://support.novell.com/linux/

  This is the home page for all Novell Linux support and includes links to support options such as the Knowledgebase, downloads, and FAQs.

- http://www.novell.com/coolsolutions

  This Novell web site provides the latest implementation guidelines and suggestions from Novell on a variety of products, including SUSE Linux.

Agenda

The following is the agenda for this five-day course:

Table Intro-1

	Section	Duration
Day 1	**Introduction**	01:00
	Section 1: Configure a DNS Server Using BIND (Part I)	05:00
Day 2	**Section 1:** Configure a DNS Server Using BIND (Part II)	01:00
	Section 2: Use DHCP to Manage Networks	03:00
	Section 3: Manage OpenLDAP (Part I)	02:00
Day 3	**Section 3:** Manage OpenLDAP (Part II)	03:00
	Section 4: Configure a Mail Server (Part I)	03:00
Day 4	**Section 4:** Configure a Mail Server (Part II)	03:00
	Section 5: Use OpenSLP	01:00
	Section 6: Monitor Network Traffic (Part I)	02:00
Day 5	**Section 6:** Monitor Network Traffic (Part II)	02:00
	Appendix A: Prepare for the Novell CLE Practicum	04:00

Scenario

The Digital Airlines management has made the decision to use network services shipped with SUSE Linux Enterprise Server 10, including BIND, DHCP, OpenLDAP, mail server, and OpenSLP.

You have already installed SUSE Linux Enterprise Server 10 and you are familiar with administering SUSE Linux Enterprise Server 10 from YaST and from the command line.

To deploy the services as planned, you need experience in the following areas:

- Domain name system (DNS)
- Dynamic host configuration (DHCP)
- OpenLDAP
- Email management
- OpenSLP
- Network traffic monitoring

You decide to set up test servers in the lab so you can practice your skills in these areas.

Exercises

The exercises in this course consist of a description of the exercise, and step-by-step instructions on how to solve the task.

You should first try to solve the task described on you own, based on what is covered in the manual in the respective section. Resort to the step-by-step instruction only if you feel unable to solve the task or to find out if what you did was correct.

Exercise Conventions

When working through an exercise, you will see conventions that indicate information you need to enter that is specific to your server.

The following describes the most common conventions:

- ***italicized/bolded text***. This is a reference to your unique situation, such as the host name of your server.

  For example, if the host name of your server is DA50, and you see the following

  ***hostname*.digitalairlines.com**

  you would enter

  DA50.digitalairlines.com

- **10.0.0.*xx***. This is the IP address that is assigned to your SUSE Linux Enterprise Server 10 server.

  For example, if your IP address is 10.0.0.50, and you see the following

  10.0.0.*xx*

  you would enter

  10.0.0.50

- **Select.** The word ***select*** is used in exercise steps to indicate a variety of actions including clicking a button on the interface and selecting a menu item.
- **Enter and Type.** The words ***enter*** and ***type*** have distinct meanings.

  The word ***enter*** means to type text in a field or at a command line and press the Enter key when necessary. The word ***type*** means to type text without pressing the Enter key.

  If you are directed to type a value, make sure you do not press the Enter key or you might activate a process that you are not ready to start.

SECTION 1 Configure a DNS Server Using BIND

Computers communicate with each other using their IP addresses. But for humans, it is simpler to address a computer using its name.

DNS (Domain Name System) is a distributed database system that guarantees unique computer names worldwide. It provides computers with IP addresses when a user enters a computer name, and vice versa.

Objectives

1. Understand the Domain Name System
2. Install and Configure the BIND Server Software
3. Configure a DNS Server
4. Configure the DNS Clients
5. Forward DNS Requests
6. Configure Access, Logging, and Security
7. Configure Dynamic DNS
8. Use YaST to Configure BIND
9. Monitor the DNS Server
10. Find More Information About DNS

Objective 1 Understand the Domain Name System

To understand the basics of name resolution with DNS, you need to know the following:

- How Name Resolution Worked in the Early Days of the Internet
- The Internet Domain Concept
- How Name Servers Work
- How to Query DNS

How Name Resolution Worked in the Early Days of the Internet

Computers communicate with each other by using IP addresses, but for humans it is more simple to address a computer by using its name. This requires some kind of conversion that provides computers with IP addresses when a user enters a computer name.

In the early days of the Internet, when there were relatively few computers connected to each other, a file was maintained at the Network Information Centre (NIC) of the Stanford Research Institute in California that provided exactly this conversion.

Whenever system administrators added a new computer to the Internet or changed the name of an already connected computer, these changes were sent by email to the SRI-NIC where they were written to a file called hosts.txt.

Every system administrator worldwide had to copy this file by FTP and distribute it to all computers for which he was responsible.

This procedure had several weak points:

- **Load**. Requests to the SRI-NIC created considerable network traffic. In addition the computer on which the file was located soon became completely overloaded.
- **Name collisions**. Each computer name could be assigned only once worldwide. Although the NIC was able to assign unique IP addresses, it had no influence over the choice of names.
- **Consistency**. It was very difficult to guarantee the consistency of a file which was distributed worldwide. When a version of the file HOSTS.TXT reached a computer, it could already be out of date.
- **Scalability**. As the number of computers grew, maintenance work became much greater.

In 1984, Paul Mockapetris created a powerful solution for these problems: the Domain Name System (or DNS). DNS is a distributed database system that allows local administration of areas and guarantees unique computer names worldwide. Its hierarchical structure is very similar to the tree structure of the Linux file system.

The Internet Domain Concept

DNS consists of several domains that can be divided into subdomains. The top level of this structure is the root domain. It is represented simply by a dot (“.”). There are 13 computers worldwide (actually, there are more than 13 computers to provide load balancing and fail-safe systems). The first layer below the root domain is built by the top-level domains (TLDs).

In the early days of DNS there were seven TLDs:

- **.com** for commercial institutions (such as novell.com and suse.com)
- **.edu** for educational institutions and research institutes (such as harvard.edu and stsci.edu)
- **.gov** for institutions of the U.S. government (such as nasa.gov and whitehouse.gov)
- **.int** for international institutions (such as un.int and ecb.int)
- **.mil** for military institutions (such as army.mil and navy.mil)
- **.net** for institutions that provide and manage network infrastructure (such as internic.net and att.net)
- **.org** for noncommercial institutions (such as eso.org and eff.org)

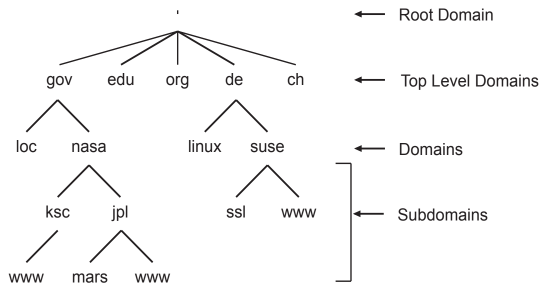
.arpa was used as a TLD, while the ARPAnet transferred from host files to DNS. The Advanced Research Projects Agency Network (ARPAnet) developed by ARPA of the U.S. Department of Defense was the world's first operational packet switching network, and the proegnitor of the global Internet. All computers from the ARPAnet were later put into the other TLDs. The arpa TLD still has a special meaning which will be explained later in this section.

These TLDs are also known as generic TLDs. Other TLDs for individual countries, such as .de for Germany, .uk for the United Kingdom, and .ch for Switzerland, were defined later.

Recently, TLDs such as .info or .biz have become operational. Each of these TLDs is administered by its own institution (the Network Information Center or NIC).

Part of the Internet namespace is shown in the following:

Figure 1-1



The complete computer name or fully qualified domain name (FQDN) is made from the actual computer name, the domain name, and the name of the TLD (one or more subdomains might be included).

Examples of FQDNs are ns.suse.de, www.astro.physik.uni-goettingen.de and mail.novell.com. To be precise, all these names end with a dot (such as ns.suse.de.) indicating the root domain. But as a rule the dot normally is not used. All applications know how to handle this.

How Name Servers Work

Domains are administered locally instead of using a global authority. Each domain has its own administration point (in practice, many domains are administered from one location).

For each domain there is one DNS server (or name server) defined as being "in charge" of its domain. This server is known as the *master server*, and it is the authority for this domain (providing authoritative answers).

This authoritative information is important because DNS servers also temporarily store information on other domains in a cache and can pass this information on, with the note that it is a non-authoritative answer.

There are other DNS servers called *slave servers* for the domain that distribute the load and serve as backups. Slave servers keep a copy of the information on the master server and update this information at regular intervals. This update is called *zone transfer*.

The following describes the DNS server types available:

Table 1-1

Master server	Has the main responsibility for a domain. Gets its data from local files.
Slave server	Gets its data from the master server using zone transfer.
Caching-only server	Queries data from other DNS servers and stores the information in the cache until its expiration date. All replies are nonauthoritative.
Forwarding server	All queries the server cannot answer authoritatively are forwarded to other DNS servers.

How to Query DNS

Various programs are involved in processing a request to the DNS database. The first is the *resolver*. This is a set of library routines used by various programs.

The resolver makes a request to a DNS server, interprets the answer (real information or error message), and sends back this information to the program that called it up.

If the DNS server receives a request from a resolver, one of two things happens:

- If the DNS server is the authority for the requested domain, the DNS server provides the required information to the resolver (the authoritative answer).

  or

- If the DNS server is not the authority for the required domain, the DNS server queries the responsible authority for the request domain and gives the result to the resolver.

  The data is stored in the cache of the DNS server. If there is another request for this data later, the DNS server can provide it immediately (a non-authoritative answer). All data has a timestamp and time-to-live (TTL) information. When the TTL expires, the information is deleted from the cache.

Assume that your DNS server wants to find the IP address of the computer www.suse.de. To do this, the DNS server first makes a request to one of the DNS servers of the root domain.

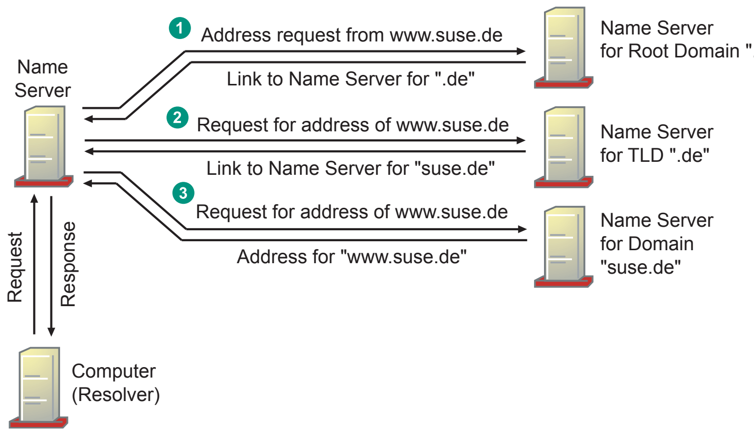
Each DNS server knows the server of the root domain. E.g., for www.suse.de, this is a DNS server for the TLD .de, that is, the computer dns2.denic.de.

Our DNS server then asks dns2.denic.de for the authority for the domain suse.de and as an answer is given the computer ns.suse.de (one of the DNS servers).

In a third step, this DNS server is queried and returnes the IP address of the SUSE web server. This answer is returned by our DNS server to the requesting resolver.

This procedure is illustrated in the following figure:

Figure 1-2



The DNS servers for the root domain play a very important role in name resolution. In order to alleviate the server load due to queries, every DNS server stores the information received from other name servers in its cache. When queries are made, this information is sent without querying the root DNS server anew. However, root DNS servers are very busy despite this caching mechanism. Several thousand queries per second are nothing unusual.

Objective 2 Install and Configure the BIND Server Software

To run a DNS server, you need to install the following packages:

- **bind.** The BIND server software (version 9.3 in SUSE Linux Enterprise Server 10)
- **bind-chrootenv.** Chroot environment for BIND server
- **bind-utils.** Utilities to query and test BIND (included in standard installation)

Before starting the DNS server, you have to make some basic configuration changes. After finishing your configuration, you can start the server using the following command:

rcnamed start

To stop a running server, use the following command:

rcnamed stop

To have the DNS server start automatically at boot time, use the following command:

insserv named

This creates the necessary links in the runlevel directories.

By default, the name server runs in a chroot environment (**NAMED_RUN_CHROOTED="yes"** in /etc/sysconfig/named).

For security reasons, chroot should always be used.

All relevant files (if they are specified in /etc/sysconfig/named) are copied to the respective subdirectories in /var/lib/named/. For example, the BIND configuration file /etc/named.conf is copied to /var/lib/named/etc/named.conf. To disable the chroot environment, set the variable to **no**.

Objective 3 Configure a DNS Server

The configuration of each DNS server type is slightly different. In this objective the following is discussed:

- Configure a Caching-Only DNS Server
- Configure a Master Server for Your Domain
- Configure One or More Slave Servers

Configure a Caching-Only DNS Server

A caching-only DNS server does not manage its own databases but merely accepts queries and forwards them to other DNS servers. The replies are saved in the cache.

The DNS server configuration is defined in the file /etc/named.conf. You can use the example file that is installed with the DNS package as a configuration file for a caching-only server.

The following example shows the beginning of a simple configuration:

```
#
# /etc/named.conf: Configuration of the name server (BIND9)
#
# Global options
#
options
{
#
# In which directory are the database files?
#
    directory "/var/lib/named";
};
```

Lines beginning with a hash sign ("#") are comments and will be ignored.

The global options are defined in the **options** block at the beginning of the file. The **directory** option specifies the directory where the database files (or zone files) are located. Normally, this is /var/lib/named/.

Using the **directory** option you can specify filenames for the database files with a relative path (no absolute path required).

The global options are followed by the definition of the database files for the domains managed by the DNS server. Several entries are needed for basic DNS server functions such as those provided by a caching-only server.

Three entries are needed for every DNS server:

- The entry for root DNS servers (not needed for BIND 9 because it has the list of root DNS servers compiled into the software).
- The forward resolution for localhost
- The reverse resolution for the network 127.0.0.0 (localhost)

The following are examples of these entries:

```
## entry for root nameservers#
zone "." in {
    type hint;
    file "root.hint";
};

#
# forward resolution for localhost
#
zone "localhost" in {
    type master;
    file "localhost.zone";
};

#
# reverse resolution for localhost
#
zone "0.0.127.in-addr.arpa" in {
    type master;
    file "127.0.0.zone";
};
```

The zone entry for the root DNS servers contains a reference to a file containing the addresses of the root DNS servers. This file (root.hint) is generated in the directory /var/lib/named/ during the installation of the package bind.

The two files for the resolution of localhost are also generated during the installation. The structure of these files is explained later.

Starting each request for name resolution with queries to the root servers can be quite slow. If these reqests are forwarded to a name server with a lot of information in its cache (e.g., the name server of your Internet service provider), the process will be much faster in most cases.

You can define these DNS servers in the options block in the file /etc/named.conf, as in the following:

```
options
{
    directory "/var/lib/named";

    forwarders {
        10.0.0.254;
    };
};
```

You can enter up to three DNS server addresses. Queries that cannot be resolved by the local DNS server are forwarded to one of the specified DNS servers.

If these DNS servers cannot be reached, the queries are sent directly to the root DNS servers.

Configure a Master Server for Your Domain

The following are the tasks you need to do to configure a master DNS server for your domain:

- Adapt the Main Server Configuration File
- Create the Zone Files
- Create Additional Resource Records

Adapt the Main Server Configuration File

You can adapt the configuration of the caching-only DNS server to configure a DNS server containing its own information files.

This configuration already contains the global entries for the directory and the forwarders entries (which can be omitted) in the options block. There are also some more options marked as comment. The file also contains the mandatory entries for the root servers and the resolution of localhost.

The global options are followed by definitions for the database files (or zone files) for the domains this DNS server serves. At least 2 files are necessary for each domain:

- A file for forward resolution (allocating an IP address to a computer name)
- A file for reverse resolution (allocating a computer name to an IP address)

If several subnets belong to a domain, then one file for each of these networks must be created for reverse resolution.

Each definition begins with the instruction **zone** (this is why the database files are also known as zone files), followed by the name of this zone.

For forward resolution, this is always the domain name. For reverse resolution, the network prefix of the IP address must be given in reverse order (10.0.0. becomes 0.0.10.) to which the suffix in-addr.arpa is added (0.0.10.in-addr.arpa).

The zone name is always followed by an “in” for Internet. (DNS servers can administer information on different name spaces, not only that of the Internet. Other name spaces are practically never used.)

The text in curly brackets defines the type of DNS server type (e.g., master) for this zone and the name of the zone file.

The entries for a domain digitalairlines.com and a network 10.0.0.0/24 would look like this:

```
#
# forward resolution for the domain digitalairlines.com
#
zone "digitalairlines.com" in {
    type master;
    file "master/digitalairlines.com.zone";
};
#
# reverse resolution for the network 10.0.0.0
#
zone "0.0.10.in-addr.arpa" in {
    type master;
    file "master/10.0.0.zone";
};
```

Create the Zone Files

The two files for the domain localhost and the file for the root DNS servers are always included in the installation. You do not need to change these files; however, you must create the files required for the actual domain.

The subdirectory /var/lib/named/master/ is used for the database files of a master server.

You need to know the following to manually create the zone files:

- Structure of the Files
- The File /var/lib/named/master/digitalairlines.com.zone
- The File /var/lib/named/master/10.0.0.zone
- The File /var/lib/named/master/localhost.zone
- The File /var/lib/named/master/127.0.0.zone

Structure of the Files

Each of the database files consists of a series of entries, or resource records. The syntax of these records is always as follows:

reference* [*TTL*] *class type value

The following describes each part of a record:

- **reference.** The reference to which the record refers. This can be a domain (or subdomain) or a standalone computer (name or IP address).
- **TTL.** The Time To Live value for the record. If this is not present, a default TTL value is used. This determines how long other name servers store this information in their cache.
- **class.** The class of the record. For TCP/IP networks, this is always IN (internet).
- **type.** The type of the record. The most important types are listed in the table below.
- **value.** The value of the record. The value depends on the type of record as listed in the following:

Table 1-2

Record Type	Meaning	Value
SOA	Start of Authority (term for the authority)	Parameters for the domain
NS	Name server	Name or IP address of one of the DNS servers for this domain
MX	Mail exchanger	Name (or IP address) and priority of a mail server for this domain
A	Address	IP address of a computer
PTR	Pointer	Name of a computer

Table 1-2 *(continued)*

Record Type	Meaning	Value
CNAME	Canonical name	Alias name for a computer

Individual entries must always start in the first column with the reference. If an entry does not start in the first column, the reference is taken from the previous entry.

The File /var/lib/named/master/digitalairlines.com.zone

Unlike earlier versions of BIND, BIND 9 requires you to specify a default TTL for all information at the beginning. This value is used whenever the TTL has not been explicitly given for an entry.

You define the TTL with the following instruction:

```
;
; definition of a standard time to live, here: two days
;
$TTL 172800
```

In this file, the semicolon is used as a comment sign.

In this example, the TTL is given in seconds. It can be given in other units as well, such as 2D for two days. Other units are M (minutes), H (hours), and W (weeks).

This is followed by the definition of the SOA (Start of Authority) entry, which specifies which DNS server has the authority for this domain:

```
;
; SOA Entry
;
digitalairlines.com. IN SOA da1.digitalairlines.com.
hostmaster.digitalairlines.com. (
     2004092601; serial number
     1D        ; refresh (one day)
     2H        ; retry (two hours)
     1W        ; expiry time (one week)
     3H        ; "negative" validity (three hours)
)
```

The domain to which this entry refers (in the example, **digitalairlines.com**) is listed first. The domain name must end with a dot. If a name does not have a trailing dot, the name of the domain is added, which would lead to an error here.

The name of the DNS server is listed after the SOA entry (in this example, **da1.digitalairlines.com** with a dot at the end). Alternatively, you could write da1, and the domain name digitalairlines.com would be added after the name.

Next comes the email address of the person who is responsible for the administration of the DNS server. The “@” usually used in email addresses must be replaced by a dot (so the email address in this example is hostmaster.digitalairlines.com). This is necessary because “@” has a special meaning as an abbreviation.

It is advisable to use a generic email address here (e.g., hostmaster@digitalairlines.com) instead of an individual email address.

The next entry is a serial number. Any number can be used, but normally the date and a version number are used here. After any change to the data in this file, the serial number has to be increased.

Slave servers use this number to determine whether or not they need to copy this zone file. If the serial number on the master server is greater than that on the slave server, the file is copied.

The serial number is followed by time information (the first three entries listed here are only important for slave servers):

- The first entry causes a slave server to query a master server after this length of time, to see if there is a new version of the file (in the example, this is 1D or one day).
- If the slave server cannot reach the master server, the next time entry specifies at what intervals new attempts should be made (in the example, this is 2H or two hours).
- If the master server is not reached for a longer period of time, the third time entry specifies when the slave server should discard its information on this zone (in the example, this is 1W or a week).

  The basic idea here is that it is better not to pass on any information than to pass on outdated information.
- The fourth entry defines for how long negative responses from the DNS server are valid. Each requesting server stores responses in its cache, even if a computer name could not be resolved (in the example, this is 3H or 3 hours).

These time definitions are followed by the name or IP address of the computer that is acting as the DNS server for this domain. In all cases, the master server must be entered here. If slave servers are used, they should also be entered, as in the following:

```
;
; entry for the name server
;
digitalairlines.com.    IN NS    da1.digitalairlines.com.
                        IN NS    da2.digitalairlines.com.
```

The name of the domain can be omitted at this point. Then the name from the previous entry (the SOA entry) is taken.

At the end of this file are the IP addresses that are allocated to computer names. This is done with A (address) entries, as in the following:

```
;
; Allocation of IP addresses to host names
;
da10                IN A    10.0.0.10
da12                IN A    10.0.0.12
da13                IN A    10.0.0.13
...
da1                 IN A    10.0.0.254
da2                 IN A    10.0.0.2
```

The File /var/lib/named/master/10.0.0.zone

The file for reverse resolution contains similar entries as the file for forward resolution. At the beginning of the file there is the definition of a default TTL and an SOA entry.

In the SOA and NS entries, the IP address of the network is written in reverse order:

```
; Database file for the domain digitalairlines.com:
; reverse resolution for the network 10.0.0.0
;
; Definition of a default TTL,here: two days
;
$TTL 172800
;
; SOA entry
;
0.0.10.in-addr.arpa. IN SOA da1.digitalairlines.com.
hostmaster.digitalairlines.com. (
    2004092601; serial number
    1D        ; refresh (one day)
    2H        ; retry (two hours)
    1W        ; expiry time (one week)
    3H        ; "negative" validity(three hours)
                        )
;; Entry for the name server
;
                IN NS    da1.digitalairlines.com.
                IN NS    da2.digitalairlines.com.
```

At the end of this file are the host names that are allocated to computer names, this time with the PTR (Pointer) entry, as in the following:

```
;
; Allocation of host names to IP addresses
;
10                IN PTR  da10.digitalairlines.com.
12                IN PTR  da12.digitalairlines.com.
13                IN PTR  da13.digitalairlines.com.
14                IN PTR  da14.digitalairlines.com.
...
254               IN PTR  da1.digitalairlines.com.
2                 IN PTR  da2.digitalairlines.com.
```

The following two files must exist for the local computer. These are created automatically during installation and should not be modified.

The File /var/lib/named/master/localhost.zone

The following is an example of the file /var/lib/named/master/localhost.zone:

```
$TTL 1W
@    IN SOA    @        root (
               42       ; serial (d. adams)
               2D       ; refresh
               4H       ; retry
               6W       ; expiry
               1W )     ; minimum

     IN NS     @
     IN A      127.0.0.1
```

In this example, the "@" character is used as an abbreviation (for this reason, it must be replaced by a dot in the email address in the database files).

Using "@" instead of the domain name causes the file /etc/named.conf to be read to see for which domain this file is responsible.

In this case, it is localhost, which is also used for the name of the DNS server (this is why "@" appears many times in the file).

The File /var/lib/named/master/127.0.0.zone

In this file, the abbreviation "@" is also used. But here the computer name must be given explicitly with localhost (remember the dot at the end):

```
$TTL 1W
@    IN SOA    localhost.    root.localhost. (
               42        ; serial (d. adams)
               2D        ; refresh
               4H        ; retry
               6W        ; expiry
               1W )      ; minimum

     IN NS     localhost.
1    IN PTR    localhost.
```

Create Additional Resource Records

Apart from the resource records already discussed (SOA, NS, A, PTR), there are MX and CNAME resource records, which are used to do the following:

- Define Mail Servers for the Domain
- Assign Aliases for Computers

Define Mail Servers for the Domain

To be able to use email addresses in the form geeko@digitalairlines.com, the email server responsible for the domain must be defined (the email cannot be sent directly to the domain, but must be sent to a mail server).

To achieve this, an MX (Mail Exchange) entry must be made in the database file for forward resolution, after the DNS server entry:

```
digitalairlines.com.    IN MX   0 mail
                        IN MX  10 da1
                        IN MX  10 da5
```

If an email is now sent to the address geeko@digitalairlines.com, the computer sending the mail asks the DNS server which computer is the mail server, and is sent the list of the MX entries in return.

Several mail servers can be given. On the basis of their priorities, it is then decided to which computer the email is sent. The priority of mail servers is defined by the number next to MX; the lower this number, the higher the priority.

In this example the computer mail.digitalairlines.com has the highest priority (it is, therefore, the primary mail server). da1.digitalairlines.com and da5.digitalairlines.com both have the same priority.

If the mail server with the highest priority cannot be reached, the mail server with the second-highest priority is used. If several mail servers have the same priority, then one of them is chosen at random. An address entry must be made for each mail server.

Assign Aliases for Computers

If you want a computer to be reached by more than one name (such as addressing a computer as da30.digitalairlines.com and www.digitalairlines.com), then corresponding aliases must be given.

These are the CNAME (canonical name) entries in the database file for forward resolution:

```
da30            IN A      10.0.0.30
www             IN CNAME  da30
```

The names of the mail servers for the domain (MX entry) cannot be alias names, since some mail servers cannot handle this correctly.

Configure One or More Slave Servers

To guarantee reliable operation, at least one more DNS server besides the master server is required. This can take over part of the load from the DNS master server. But it is especially important in case the DNS master server is not available. This additional DNS server is set up as a DNS slave server.

The essential difference between the two types is that a slave server receives copies of the zone files from the master server. Modifications to the zone files are only made on the master server.

As soon as a slave server is started, it connects to the master server and receives a copy of the zone files from it. This is called a *zone transfer.*

Comparison of data between the servers takes place automatically. On the one hand, the slave server queries the master server at regular intervals and checks, using the serial number of the zone files, whether anything has changed.

By default, the master server sends a message to all listed slave servers (called *notify*) as soon as it has been restarted in order to read in modified zone files.

In the configuration file /etc/named.conf for a slave server, there are at least two entries that define it as a master server: the two zone definitions for the loopback network (localhost).

There might also be a zone definition for the root DNS server. But a zone definition is only necessary if the slave server will forward requests to other DNS servers.

The definitions for zones for which it should copy data from the master server look like the following:

```
zone "digitalairlines.com" in {
    type slave;
    file "slave/digitalairlines.com.zone";
    masters {
        10.0.0.254;
    };
};
```

The slave server gets data from the master server with the IP address 10.0.0.254 and stores it in the directory /var/lib/named/slave/. This directory is created when you install the BIND package.

A similar configuration must be made for reverse resolution, as in the following:

```
zone "0.0.10.in-addr.arpa" in {
    type slave;
    file "slave/10.0.0.zone";
    masters {
        10.0.0.254;
    };
};
```

In the simplest configuration, the slave server gets information from the master server at regular intervals. This can cause the slave server to provide outdated information for a certain length of time.

This is why it is reasonable to instruct the master server to inform the slave servers about modifications in the database files. The slave servers then immediately carry out a zone transfer, which always brings them up to date.

In order for the master server to be able to communicate with the slave servers, it must know about them. By default, the master server automatically informs its slave servers. This can also be defined explicitely in the options section of the file /etc/named.conf, as in the following:

```
options {
    ...
    notify yes;
};
```

Subsequently, the slave servers must be listed as DNS servers in the database files (for the forward and reverse resolution):

```
digitalairlines.com.     IN NS    da1.digitalairlines.com.

                         IN NS    da8.digitalairlines.com.
```

This informs the slave server, da8.digitalairlines.com, about all modifications.

Objective 4 Configure the DNS Clients

To instruct a computer to use a certain DNS server, the information about this server has to be written to the file /etc/resolv.conf. This can be done using YaST (during installation or at any time later) or by editing this file with any text editor.

The file /etc/resolv.conf looks like the following:

```
search digitalairlines.com
nameserver 10.0.0.254
```

Normally, this file has the following two types of entries:

- **search.** A list of the names of domains (or subdomains) is provided after this keyword. Several domain names are listed on one line. This allows only the host name to be used to resolve to the correct IP address.

  The host name is expanded by the domain names specified here until a matching IP address is found.

  For example, if you provide digitalairlines.com and atl.digitalairlines.com as domain names, the host server is expanded to server.digitalairlines.com and server.atl.digitalairlines.com to look for a corresponding IP address. The first matching IP address is returned.

  If both of these host names exist, you have to specify the FQDN to resolve the IP address.

- **nameserver.** The keyword nameserver specifies the IP address of a DNS server to use. You can have up to 3 entries, but each of them must only contain 1 server address. If several entries of this type exist, the DNS servers are queried in this order.

There is another important file for the clients: /etc/nsswitch.conf. This file applies to all programs that use the resolver functions of the current GNU C Library (libc6). (The predecessor of this file is /etc/host.conf, which applies to older versions of the GNU C Library.)

This file configures the name service switch, which is responsible for resolving host names, network names, users, and groups.

The relevant part for resolving host names looks like the following:

```
#
# /etc/nsswitch.conf
#
...
hosts:          files dns
networks:       files dns
...
```

Both entries shown here define that the first attempt to resolve a host name is done using the file /etc/hosts. If this fails, a DNS server is contacted to resolve the name. The same applies to the resolution of network names, done using /etc/networks first.

Objective 5 Forward DNS Requests

If the name server does not have any information on the computer to be resolved, it tries to fetch this information from another name server.

To do this, the name server has to know how to reach other name servers:

- The Name Servers of the Root Domain
- Forward Requests to Specific Name Servers
- Forward Requests for a Subdomain

The Name Servers of the Root Domain

In the configuration file /etc/named.conf there should be a zone entry containing a reference to a file which contains the the addresses of the root name servers.

This file is created during the installation of the bind package in the directory /var/lib/named/ and is called root.hint.

The zone entry in the file /etc/named.conf looks like this:

```
zone "." in
{
    type hint;
    file "root.hint";
};
```

Forward Requests to Specific Name Servers

If no additional entries are made, the name server can contact the root name servers directly. But it is also possible to give the name server the addresses of other name servers. It will use these first if it cannot resolve a computer name itself.

The definition of such a name server is specified in the **options** section of the file /etc/named.conf:

```
options {
    directory "/var/lib/named";
    forwarders {
        10.0.0.10;
        10.0.0.15;
    };
};
```

In this case, the requests that could not be resolved locally are forwarded to one of the two name servers given. If neither of these can be reached, the request runs via the root name servers, as described above.

Forward Requests for a Subdomain

If there is another name server used for resolving host names for a subdomain, this name server has to be defined in the corresponding zone entry.

The type is set to forward, and the address of the respective name server has to be specified:

```
zone "slc.digitalairlines.com" in {
    type forward;
    forward only;
    forwarders {
        10.0.0.10;
    };
};
```

If you include the entry **forward only**, the name server is instructed never to try to resolve the address itself. All requests concerning the subdomain slc.digitalairlines.com are forwarded to the name server with the address 10.0.0.10. A corresponding instruction for reverse resolution must be defined.

Exercise 1-1 Configure a DNS Server with Forwarding

In this exercise, you work with a partner to configure a DNS master server and a DNS slave server for the domain digitalairlines.com.

You will find this exercise in the workbook.

(End of Exercise)

Objective 6 Configure Access, Logging, and Security

In the last objective you learned how to configure a DNS server. In this objective you will learn about the following configuration options:

- Restrict Access to the Name Server
- Configure Logging Options
- Security Aspects for Zone Transfer

Restrict Access to the Name Server

To prevent a name server from becoming overloaded, it can be useful to allow access only from specific computers or networks.

To do this, use the instruction **allow-query**. This instruction is defined in the **options** section of the /etc/named.conf file:

```
options {

    ...

    allow-query {
        10.0.0.0/24;
        10.0.1.0/24;
    };
};
```

If you include this instruction, only computers from the networks defined can access this name server.

All other computers will receive an error message.

Do not use this instruction for the name server that provides information about your network to hosts on the Internet.

Configure Logging Options

The configuration statement logging can be used to define the logging behavior of BIND.

Log messages are written by using channels. In addition to some predefined channels, an unlimited number of channels can be defined. Every channel contains a destination that determines what to do with the messages.

Possible destinations are

- **syslog**. Messages are passed to the syslog daemon.
- **null**. Messages are dumped.
- **file**. Specifies the file to which the messages are written. You can add further options e.g.,
  - **size**. The maximum file size. If this size is reached, a new version of the file is created.
  - **version**. Number of versions of the file that are stored.

Every message has a priority level. You can use the entry **severity** to determine from which priority level messages are written to this channel. Possible levels of **severity** are

- **critical**. Similar to the "critical" level used by syslog.
- **error**. Similar to the "error" level used by syslog.
- **warning**. Similar to the "warning" level used by syslog.
- **notice**. Similar to the "notice" level used by syslog.
- **info**. Similar to the "info" level used by syslog. This is the default severity.

- **debug [*level*]**. You can specify a debug level. Logging will occur for any message equal to or higher than the level specified; lower levels will not be logged.

  level can be an integer between 0 (no debugging logs) and 11 (full debugging log). The default level is 1.

- **dynamic**. If you specify **dynamic** severity, then the name server inherits its debug level from the server startup debug level.

  You can specify the server's startup debug level by starting named with the parameter **-d [*level*]**.

The following is an example for a channel definition labeled "sec_file":

```
logging {
    channel sec_file {
        file "/var/log/bind_security.log" versions 3 size 20m;
        severity debug 3;
        print-time yes;
        print-category yes;
        print-severity yes;
    };
};
```

Here you can see the syntax of the logging option: The **logging** option includes the definition for channel sec_file.

All messages with debug level 3 sent to this channel are written to the /var/log/bind_security.log file with a time stamp (**print-time yes**), category (**print-category yes**) and severity level (**print-severity yes**). As soon as the file reaches a size of 20 MB (**size 20m**), a new file is created. Three old versions of the file are stored (**version 3**).

The following is a further example for two channel definitions:

```
channel default_syslog {
    syslog daemon;
    severity info;
};

channel null {
    file "/dev/null";
};
```

Here, two predefined channels are listed: syslog daemon and null device.

It is possible to send logging messages to more than one channel. Use the parameter **category** to determine where messages are written to. The following is an example for a **category** entry:

```
channel "my_security_channel" {
    file "/var/log/bind_security.log";
    severity info;
};

channel default_syslog {
    syslog daemon;
    severity info;
};

category "security" {
    "my_security_channel";
    "default_syslog";
};
```

This logs security events to the file /var/log/bind_security.log but also keeps the default logging behavior.

Security Aspects for Zone Transfer

In this section we focus on two security aspects for the zone transfer:

- Use Encryption
- Zone Transfers from Slave Servers

Use Encryption

It is important to ensure that only authorized hosts can perform a zone transfer. Restricting the zone transfer is done using the **allow-transfer** statement. This statement takes either an IP address or the name of an encryption key as a parameter. We strongly recommend that you use an encryption key, since the IP address can be spoofed quite easily.

The key is used to sign requests from the slave server. Only signed requests for zone transfers are accepted by the master server. First, a key must be generated and made known to the master server and slave servers. To do this, you do the following steps:

- Create a Key
- Configure the Master Server
- Configure the Slave Server

Create a Key

To create a key, use the **dnssec-keygen** command. The file name of the key is printed on the screen:

```
da51:/var/lib/named # dnssec-keygen -a HMAC-MD5 -b 128 -n HOST
zonetransfer
Kzonetransfer.+157+01389
```

The options are explained in the following table:

Table 1-3

Option	Description
-a HMAC-MD5	The encryption procedure used (here HMAC-MD5)
-b 128	The length of the key (in the example, 128 bits)
-n HOST	The type of key
zonetransfer	Name of key

If you use this command, two files are created in the current directory:

```
da51:/var/lib/named # ls -l K*
-rw------- 1 root root 56 Feb 21 10:39 Kzonetransfer.+157+01389.key
-rw------- 1 root root 81 Feb 21 10:39 Kzonetransfer.+157+01389.private
```

The numbers at the end of the filename will be slightly different when you run this command.

- The *.key file contains a DNS KEY record that can be included in a zone file using the **include** statement.
- The *.private file contains algorithm specific fields. For security reasons, this file does not have general read permission.

Both files contain the same key:

```
da51:/var/lib/named # cat Kzonetransfer.+157+01389.key
zonetransfer. IN KEY 512 3 157 KhDVspoqFonWKv58rFXOWw==
da51:/var/lib/named # cat Kzonetransfer.+157+01389.private
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: KhDVspoqFonWKv58rFXOWw==
```

This key has to be included in the configuration file /etc/named.conf on both the master server and the slave server.

Configure the Master Server

The following entries must be made in the file /etc/named.conf on the master server:

```
options {
    ...
};

key zonetransfer {
    algorithm HMAC-MD5;
    secret "KhDVspoqFonWKv58rFXOWw==";
};

zone "digitalairlines.com" in {
    type master;
    file "master/digitalairlines.com.zone";
    allow-transfer {
        key zonetransfer;
    };
};
```

A corresponding entry must be made for reverse resolution.

```
zone "0.0.10.in-addr.arpa" in {
    type master;
    file "master/10.0.0.zone";
    allow-transfer {
        key zonetransfer;
    };
};
```

The key (the file /etc/named.conf) should not be readable for anybody except root and the group named.

This is ensured by the predefined permissions of /etc/named.conf:

```
da51:~ # ls -l /etc/named.conf
-rw-r-----  1 root named 572 Jun  3 14:30 /etc/named.conf
```

If you include the key using the **include** statement, it is not necessary to change the permissions of the file /etc/named.conf.

Configure the Slave Server

On the slave server, define the key in the /etc/named.conf file. All requests to the master server will be signed with this key (by using the **server** instruction).

The following is an example:

```
options {
    ...
};

key zonetransfer {
    algorithm HMAC-MD5;
    secret "KhDVspoqFonWKv58rFXOWw==";
};

server 10.0.0.2 {
    keys {
        zonetransfer;
    };
};
```

It is also important that only the user root and the group named be able to read the configuration file containing the key.

If a computer now tries to carry out a zone transfer without possessing the key, the zone transfer is refused. On the master server, corresponding messages are listed in the file /var/log/messages. A successful zone transfer generates a message in /var/log/messages similar to the following:

```
Apr 21 10:37:18 da51 named[3976]: client 10.0.0.20#32769: transfer of
'digitalairlines.com/IN': AXFR started
```

A failed zone transfer generates a message similar to this in /var/log/messages:

```
Apr 21 10:38:59 da51 named[3976]: client 10.0.0.20#32775: request has
invalid signature: tsig verify failure
```

In /var/log/messages on the slave server, a message similar to this is generated:

```
Apr 21 10:40:01 da20 named[4099]: zone 0.0.10.in-addr.arpa/IN: refresh:
failure trying master 10.0.0.51#53: tsig indicates error"
```

Zone Transfers from Slave Servers

You should prohibit zone transfers from slave servers although it is possible to perform them.

If you only have one server (the master) that allows zone transfers to slave servers, you must ensure that all slave servers get exactly the same information.

To prohibit the zone transfer from a server, use the following statement:

```
options {
    allow-transfer {none;};
};
```


Exercise 1-2 Configure Zone Transfers from the Master Server to Slave Servers

In this exercise, you configure zone transfers from a master to a slave server.

You will find this exercise in the workbook.

(End of Exercise)

Objective 7 Configure Dynamic DNS

If the number of zones and hosts increases, it is inconvenient to edit the zone files manually.

You can modify the resource record sources of the name server dynamically without editing and reloading files. This is called dynamic DNS. Dynamic DNS can also be used by external services like DHCP.

To allow dynamic changes, add the following line in the zone definition.

```
allow-update { 127.0.0.1; };
```

In this example, only the loopback address is used, but it is also possible to add IP addresses of other hosts.

To edit the DNS records dynamically, use the command **nsupdate**:

```
da51:~ # nsupdate
>
```

The security tool AppArmor is installed and enabled on SUSE Linux Enterprise Server 10 by default. There is an AppArmor profile for BIND that prohibids **nsupdate** to change the BIND zone information.
AppArmor is covered in the course *SUSE Linux Enterprise Server 10 Security* (Course 3075) in detail.
Stop the AppArmor daemon by entering **rcapparmor stop** to use **nsupdate** as described in the following.

Before using nsupdate, make sure that all zone files have the correct access permissions. If they are not set properly, use the following commands:
chgrp -R named /var/lib/named
chmod -R g+w /var/lib/named

Dynamic updates are stored in a journal file for the zone. This file is automatically generated by the server when the first dynamic update is performed. The name of the journal file is created by appending the extension .jnl to the name of the corresponding zone file. The journal file is in a binary format and should not be edited manually.

The contents of the journal file are written to the zone file every 15 minutes. When the name server is shut down, the contents of the journal file are written to the zone file, too.

Dynamic updates will change the layout of your zone files when the data is written to them. You should either use an editor or **nsupdate** to modify your zone information instead of using both tools.

The most important **nsupdate** options are

- **server *server* [*port*]**. Sends all dynamic update requests to the name server ***server***.

  If no ***port*** number is specified, the default DNS port number 53 is used.

- **update delete *reference* [*ttl*] [*class*] [*type* [*value*...]]**. Deletes any resource records named ***reference***.

  If the record type and value are provided, only matching resource records will be removed.

  The Internet class (IN) is assumed if ***class*** is not supplied.

  ttl is ignored. It is only allowed for compatibility.

- **update add *reference ttl* [*class*] *type value*...**. Adds a new resource record with the specified ***ttl*** (in seconds), ***class*** and ***value***.

- **send**. Sends the current message. This is necessary to actually execute the command. This is equivalent to entering a blank line.

All commands are described in the man page of nsupdate: **man 8 nsupdate**.

The following is an example of using **nsupdate**:

```
da51:~ # nsupdate
> server 127.0.0.1
> update add da13.digitalairlines.com 86400 A 10.0.0.13
>
> update delete da13.digitalairlines.com A
>
> update add 13.0.0.10.in-addr.arpa 86400 PTR da13.digitalairlines.com
>
```

This will generate messages like the following in /var/log/messages:

```
May 20 11:13:58 da51 named[5161]: client 127.0.0.1#32781: updating zone
'digitalairlines.com/IN': adding an RR
May 20 11:13:58 da51 named[5161]: journal file
master/digitalairlines.com.zone.jnl does not exist, creating it
May 20 11:13:58 da51 named[5161]: zone digitalairlines.com/IN: sending
notifies (serial 2005051903)
May 20 11:21:50 da51 named[5161]: client 127.0.0.1#32783: updating zone
'0.0.10.in-addr.arpa/IN': adding an RR
May 20 11:21:50 da51 named[5161]: journal file master/10.0.0.zone.jnl does
not exist, creating it
May 20 11:21:50 da51 named[5161]: zone 0.0.10.in-addr.arpa/IN: sending
notifies (serial 2005051902)
```

The journal files will be created automatically:

```
da51:/var/lib/named # dir master/
total 16
drwxrwxr-x  2 root  named 200 May 20 11:21 .
drwxrwxr-x  9 root  named 408 May 20 10:40 ..
-rw-rw-r--  1 root  named 463 May 19 12:06 10.0.0.zone
-rw-r--r--  1 named named 814 May 20 11:21 10.0.0.zone.jnl
-rw-rw-r--  1 root  named 440 May 19 14:51 digitalairlines.com.zone
-rw-r--r--  1 named named 794 May 20 11:13 digitalairlines.com.zone.jnl
```

Press **Ctrl + D** or enter **quit** to quit nsupdate.

In the following table, the most important record types are listed:

Table 1-4

Record Type	Meaning	Value
SOA	Start of Authority	Parameter for the domain
NS	DNS server	Name of a DNS server for this domain
MX	Mail exchanger	Name and priority of a mail server for this domain
A	Address	IP address of the computer
PTR	Pointer	Name of the computer
CNAME	Canonical name	Alias name for the computer

Alternatively, you can specify a file containing the needed commands. An ASCII file (called **updates**) with the following content delivers the same result as the interactive DNS update as shown above:

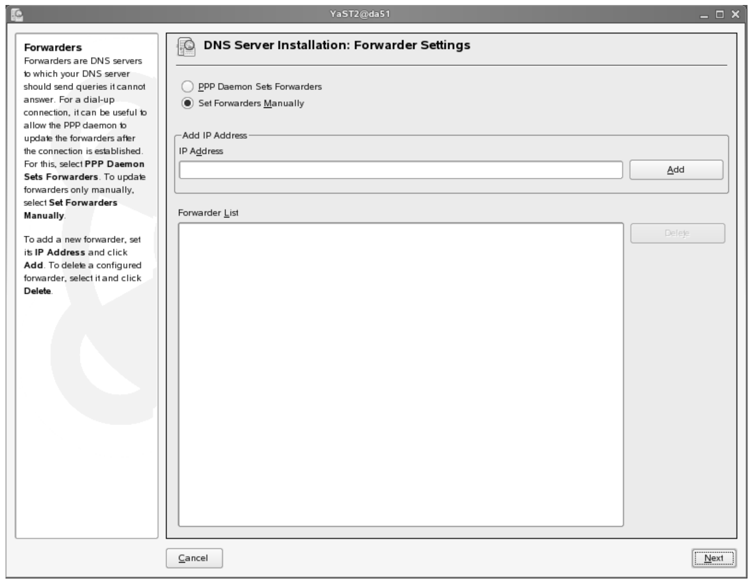
```
da51:~ # cat updates
update delete da7.digitalairlines.com
update add da8.digitalairlines.com 84600 A 10.0.0.8
da51:~ # nsupdate -v -k Kdhcp-dns.+157+23165.key updates
da51:~ #
```

Objective 8 Use YaST to Configure BIND

YaST provides a module for the configuration of a name server. To start the module, select

YaST2 > Network Services > DNS Server

Figure 1-3



YaST is only able to decode manually created configuration and zone files if the files have a very specific structure.

See the installation and administration guide for details. We want to have a closer look at another special aspect of SUSE Linux Enterprise Server 10 here.

Besides /etc/named.conf, additional configuration files can be included. By default /etc/named.conf.include is searched for. This is definded at the end of /etc/named.conf:

```
include "/etc/named.conf.include";
```

Additional configuration files can be specified in /etc/sysconfig/named. The variable **NAMED_CONF_INCLUDE_FILES** has to contain the names of the configuration files to be included. If these filenames do not contain an absolute path, they have to be located in the directory /etc/named.d/.

All relevant files (if they are specified in /etc/sysconfig/named) are copied to the respective subdirectories in the chroot envirronment (/var/lib/named/). For example, the BIND configuration file /etc/named.conf is copied to /var/lib/named/etc/named.conf.

The content of the variable **NAMED_CONF_INCLUDE_FILES** is interpreted by the start script /etc/init.d/named.

Additional options for starting the name server can be specified in /etc/sysconfig/named.

Objective 9 Monitor the DNS Server

In this objective, you learn about some tools that help to monitor your DNS server:

- Use Command Line Tools to Query DNS Servers
- Use rndc to Control the Name Server
- Check Configuration Files

Use Command Line Tools to Query DNS Servers

Several command line tools are available to query DNS server. These include the following:

- host Command
- dig Command

host Command

The most important command line tool for querying a DNS server is called **host**. The general syntax is as follows:

host *computer nameserve*r

The following example shows how it is used:

```
da51:~ # host da50
da50.digitalairlines.com has address 10.0.0.50
da51:~ # host 10.0.0.49
49.0.0.10.in-addr.arpa domain name pointer da49.digitalairlines.com.
```

If a name server address is not provided, **host** contacts the servers listed in /etc/resolv.conf. If you want to use another DNS server, you have to provide its IP address with the command.

By default, host returns the IP address or the host name, depending on which information is given. If you want to query domain information, you need to use the option **-t** with the type of information required, as in the following:

```
da51:~ # host -t ns novell.com
novell.com name server ns.novell.com.
novell.com name server ns1.westnet.net.
novell.com name server ns.utah.edu.
da51:~ # host -t mx novell.com
novell.com mail is handled by 5 prv-mx.provo.novell.com.
novell.com mail is handled by 5 prv1-mx.provo.novell.com.
novell.com mail is handled by 5 minotaur.novell.com.
da51:~ # host -t soa novell.com
novell.com has SOA record ns.novell.com. bwayne.novell.com.
2006053001 3600 900 604800 21600
```

In this example, the host names of the DNS servers for the domain novell.com are requested.

dig Command

A more verbose command is **dig**, which is normally used to troubleshoot DNS problems. The general syntax is as follows:

dig @*nameserver computer type query_options*

The options are listed in the following table:

Table 1-5

Option	Description
nameserver	The IP address or name of the DNS server that should be queried. If not specified, dig checks all DNS servers listed in /etc/resolv.conf.
computer	The resource record to query about (such as a host name, an IP address, or a domain name).

Table 1-5 *(continued)*

Option	Description
type	The type of resource record to be returned, such as a (IP address), ns (DNS server), mx (mail exchanger), x (pointer), or any (all information).
query_options	Defines how the query is done and how the results are displayed. Each query option starts with a plus sign (+).

The most important difference between host and **dig** is that dig does not use the domain list from /etc/resolv.conf by default to expand the host name. This means that the FQDN or IP address of the host must be specified. If the domain list should be used, you need to use the query option **+search**.

The following example demonstrates the use of **dig**:

```
da51:~ # dig ripe.net ns

; <<>> DiG 9.2.3 <<>> ripe.net ns
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1315
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0,
ADDITIONAL: 9

;; QUESTION SECTION:
;ripe.net.                      IN      NS

;; ANSWER SECTION:
ripe.net.        158814  IN      NS      ns2.nic.fr.
ripe.net.        158814  IN      NS      sunic.sunet.se.
ripe.net.        158814  IN      NS      auth03.ns.uu.net.
ripe.net.        158814  IN      NS      munnari.oz.au.
ripe.net.        158814  IN      NS      ns.ripe.net.

;; ADDITIONAL SECTION:
ns.ripe.net.     171939  IN      A       193.0.0.193
ns.ripe.net.     171939  IN      AAAA    2001:610:240:0:53::193
ns2.nic.fr.      344302  IN      A       192.93.0.4
ns2.nic.fr.      344302  IN      AAAA    2001:660:3005:1::1:2
sunic.sunet.se.  172586  IN      A       192.36.125.2
auth03.ns.uu.net.170436  IN      A       198.6.1.83
munnari.oz.au.   170107  IN      A       128.250.22.2
munnari.oz.au.   170107  IN      A       128.250.1.21
munnari.oz.au.   21410   IN      AAAA    2001:388:c02:4000::1:21

;; Query time: 51 msec
;; SERVER: 10.0.0.254#53(10.0.0.254)
;; WHEN: Mon Sep 27 15:27:01 2004
;; MSG SIZE  rcvd: 329
```

The QUESTION SECTION shows what was queried and the ANSWER SECTION shows the response: a list of DNS servers of the domain ripe.net.

The IP addresses of certain DNS servers are listed under ADDITIONAL SECTION. The address in the last line is an IPv6 address (**2001:388:c02:4000::1:21**).

Data about the query, such as the duration of the query (**Query time**), the server that answered the query (**SERVER**), and the date of the query (**WHEN**) is listed at the end of the output.

Use rndc to Control the Name Server

You can use the tool **rndc** (remote name daemon control) to manage the name server from a local or a remote host.

To use this tool, the requests have to be authenticated. This is done using a key similar to the key for the zone transfer. By default, a key is provided in the file /etc/rndc.key.

Because this file comes with a default key from the bind package, you should not use this key on a production system. You should always create your own key.

The key for rndc is generated using the command **rndc-confgen**. This generates a default configuration that is sent to standard output.

```
da51:~ # rndc-confgen > /etc/rndc.conf
```

The configuration file /etc/rndc.conf on the client used to manage the name server looks like this:

```
key rndc_key {
    algorithm "HMAC-MD5";
    secret "d3YWEw9jxo5UZ6N/EcIbFg==";
};

options {
    default-key “rndc-key”
    default-server 10.0.0.2;
    default-port 953;
};
```

The first statement defines the algorithm used to generate the key and the secret key. The second statement defines the default server to manage the request (the IP address of the name server) and the default key to use for the requests.

Using rndc, you can always provide other parameters.

Because the file /etc/rndc.conf contains the key, read access needs to be limited:

```
da51:~ # ls -l /etc/rndc.conf
-rw-r-----  1 root named 141 Apr  7 16:16 /etc/rndc.conf
```

Instead of creating /etc/rndc.conf and adding information about the key to /etc/named.conf, the file /etc/rndc.key can be used for automatic rndc configuration. The file /etc/rndc.key is created using the command **rndc-confgen -a**.

On the name server, access via rndc has to be permitted. This is done in the file /etc/named.conf by using the **controls** statement.

You also have to provide information about the key.

```
key rndc_key {
    algorithm HMAC-MD5;
    secret "d3YWEw9jxo5UZ6N/EcIbFg==";
};

controls {
    inet 127.0.0.1
        allow {localhost;} keys {rndc_key;};
    inet 10.0.0.2
        allow {localhost; 10.0.0.51;} keys {rndc_key;};
};
```

The **inet** statement defines which computers get access using rndc. The first parameter defines the IP address on which named will listen for rndc requests.

In this example, the first statement instructs the computer to listen on the loopback interface with the IP address 127.0.0.1, and the second statement instructs the computer to listen on the interface with the IP address 10.0.0.2.

After the **allow** statement, a list of hosts that are allowed to access is provided. For the loopback interface, access is allowed only from localhost. For the IP address 10.0.0.2, access is allowed from local host and computer 10.0.0.51. In both cases, the key named rndc_key has to be used to get access.

When you add the controls statement in /etc/named.conf, the name server listens on port 953 after reloading:

```
da51:~ # nmap da1

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2005-04-21 12:34
CEST
Interesting ports on da2.digitalairlines.com (10.0.0.2):
(The 1653 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
22/tcp  open  ssh
53/tcp  open  domain
111/tcp open  rpcbind
427/tcp open  svrloc
631/tcp open  ipp
953/tcp open  rndc
```

If you enter rndc without any options, you see a list of available parameters:

```
da51:~ # rndc
Usage: rndc [-c config] [-s server] [-p port] [-k key-file ] [-y key] [-V]
command

command is one of the following:

  reload        Reload configuration file and zones.
  reload zone [class [view]]
                Reload a single zone.
  refresh zone [class [view]]
                Schedule immediate maintenance for a zone.
  reconfig      Reload configuration file and new zones only.
  stats         Write server statistics to the statistics file.
  querylog      Toggle query logging.
  dumpdb        Dump cache(s) to the dump file (named_dump.db).
  stop          Save pending updates to master files and stop the server.
  halt          Stop the server without saving pending updates.
  trace         Increment debugging level by one.
  trace level   Change the debugging level.
  notrace       Set debugging level to 0.
  flush         Flushes all of the server's caches.
  flush [view]  Flushes the server's cache for a view.
  status        Display status of the server.
  *restart      Restart the server.

* == not yet implemented
Version: 9.2.3
```

Using the options, you can always specify another server or another key to use for the request.

The following is an example for using rndc:

```
da51:~ # rndc status
number of zones: 4
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
server is up and running

da51:~ # rndc dumpdb
```

The last line creates a dump of the cache of the name server. The dump file is defined in /etc/named.conf with the statement dump_file. By default it is /var/lib/named/log/named_dump.db. This file needs to be created with the correct permissions before cache data can be written to it:

- **touch /var/lib/named/log/named_dump.db**
- **chgrp named /var/lib/log/named_dump.db**
- **chmod g+w /var/lib/log/named_dump.db**

You can view the contents of the dump file using less or cat.

If an rndc request is tried from a host without access permissions (either not listed in the controls statement or without the correct key), an error message similar to the following will appear in /var/log/messages on the name server:

```
Apr 21 11:29:28 da51 named[4210]: rejected command channel message from
10.0.0.21#32770
```

On the denied client, the following message displays:

```
da51:~ # rndc dumpdb
rndc: recv failed: connection reset
```

Check Configuration Files

If you changed the configuration of your name server, check the syntax of the configuration files before you enable them.

To change the configuration, the BIND package provides tools:

- named-checkconf
- named-checkzone

named-checkconf

named-checkconf checks the syntax of the named configuration file (/etc/named.conf by default).

If the tool cannot detect an error, there is no output.

In case of an error, you get a message including the line number and a hint:

```
da51:~ # named-checkconf
/etc/named.conf:52: expected IP address near 'listen-on-v6'
```

If you want to check another file instead of /etc/named.conf**,** you can append the filename to the named-checkconf command.

named-checkzone

The command **named-checkzone** checks the syntax of a zone file.

named-checkzone has two parameters:

named-checkzone *zone-name file-name*

In contrast to named-checkconf, you see a message if no errors are found:

```
da51:~ # named-checkzone digitalairlines.com
/var/lib/named/master/digitalairlines.com.zone
zone digitalairlines.com/IN: loaded serial 2005071501
OK
```


Objective 10 Find More Information About DNS

If there are syntax errors in one of the configuration or zone files, BIND writes verbose messages to the file /var/log/messages. These messages also contain information on the filename and the line in which this error occurs.

If there is an error, the processing of the file is interrupted at this point (that is, errors later in the file are not detected now).

For more information about BIND and DNS, see *DNS and BIND* by Paul Albitz and Cricket Liu and the BIND homepage at http://www.isc.org/sw/bind/.

Summary

Objective	Summary
1. Understand the Domain Name System	DNS translates host names into IP addresses. DNS is a distributed database. On SUSE Linux Enterprise Server 10, you can use the BIND software to set up your own DNS server. A caching-only DNS server is not responsible for a domain, it just forwards requests to other name servers and caches the result for later requests. A master server is responsible for a domain. It provides authoritative information about this domain. DNS server information is stored in zone files. A slave DNS server receives copies of the domain zone files from the master server. Using slave servers enhances the reliability of the DNS.
2. Install and Configure the BIND Server Software	To run a DNS server, you need to install the following packages: ■ **bind.** The BIND server software (version 9.3 in SUSE Linux Enterprise Server 10) ■ **bind-utils.** Utilities to query and test BIND (included in standard installation) ■ **bind-chrootenv**. Chroot environment for bind

Objective	Summary
3. Configure a DNS Server	The configuration of the different DNS server types is slightly different. In this objective the following is discussed: ■ Configure a Caching-Only DNS Server ■ Configure a Master Server for Your Domain ■ Configure One or More Slave Servers The following are the tasks you need to do to configure a master DNS server for your domain: ■ Adapt the Main Server Configuration File ■ Create the Zone Files ■ Create Additional Resource Records

Objective	Summary
4. Configure the DNS Clients	You simply have to enter the IP address of the DNS server and possibly add some information about your domain if you are using YaST to set up your DNS client. This information is written to the file /etc/resolv.conf. /etc/resolv.conf has the following two types of entries: ■ **search.** A list of the names of domains (or subdomains) is provided after this keyword. ■ **nameserver.** The keyword nameserver specifies the IP address of a DNS server to use. /etc/nsswitch.conf configures the name service switch, which is responsible for resolving host names, network names, users, and groups.

Objective	Summary
5. Forward DNS Requests	If the name server does not have any information on the computer to be resolved, it tries to fetch this information from another name server. In the configuration file /etc/named.conf, there should be a zone entry containing a reference to a file that contains the IP addresses of the root name servers. In this case the requests that could not be resolved locally are forwarded to one of the root servers. If requests for a subdomain are forwarded to another name server, that server must be defined in the corresponding zone entry. Using the entry **forward only**, the name server is instructed never to try to resolve the address itself.

Objective	Summary
6. Configure Access, Logging, and Security	To prevent a name server from becoming overloaded, allow access only from specific computers or networks. This instruction is defined in the **options** section of the /etc/named.conf file. The logging block in the file /etc/named.conf contains the configuration of the BIND logging options. Log messages are written by means of channels. In addition to some predefined channels, an unlimited number of channels can be defined. Each channel has to contain a destination that determines what to do with the messages. Ensure that only authorized hosts can perform a zone transfer. This can be done by using an encryption key. Generate the key by using the **dnssec-keygen** command.

Objective	Summary
7. Configure Dynamic DNS	You can to modify the resource records of BIND dynamically and without editing and reloading files. Dynamic DNS can also be used by external services like DHCP. To manipulate the DNS records dynamically, use the **nsupdate** command.
8. Use YaST to Configure BIND	A BIND 9 name server can also be configured by selecting **YaST2 > Network Services > DNS Server** The last entry of the /etc/named.conf file (installed with the bind package) interprets the file /etc/named.conf.include. This meta file contains all files to be interpreted in addition to the /etc/named.conf file.

Objective	Summary
9. Monitor the DNS Server	**host** contacts the servers listed in /etc/resolv.conf. If you want to use another DNS server, you have to provide its IP address with the command. The most important difference between **host** and **dig** is that dig does not use the domain list from /etc/resolv.conf by default to expand the host name. The tool **rndc** (remote name daemon control) can be used to manage the name server from a local or a remote host. You can create your key by using the command **rndc-confgen**. The configuration file /etc/rndc.conf on the client needs the following information: ■ The algorithm used for the generation of the key ■ The IP address of the name server ■ The default key to use for the requests **named-checkconf** checks the syntax of the named configuration file (/etc/named.conf by default). **named-checkzone** checks the syntax of a zone file.
10. Find More Information About DNS	If there are syntax errors in one of the configuration or zone files, BIND writes verbose messages to the file /var/log/messages.

SECTION 2 Use DHCP to Manage Networks

DHCP (Dynamic Host Configuration Protocol) is based on BOOTP (Bootstrap Protocol). BOOTP is used to provide network configuration parameters for hosts during startup.

During the boot process, the client sends a request via broadcast into the local network. A BOOTP server answers and sends the configuration parameters (at least the IP address of the client).

DHCP dynamically assigns network configuration parameters to computers. This includes IP address, network mask, broadcast address, default gateway, DNS server to be used, and others.

DHCP is more flexible than BOOTP. For example, DHCP uses a lease time, so the delivered network information is valid only for a certain time. DHCP can also create pools of limited network addresses that can be shared in the network.

Objectives

1. Understand How DHCP Works
2. Configure the DHCP Server
3. Configure DHCP Clients
4. Configure a DHCP Relay Server
5. Use DHCP and Dynamic DNS
6. Configure DHCP Failover
7. Troubleshoot DHCP

Objective 1 Understand How DHCP Works

Every network card is uniquely identified by a 6-byte serial number, called the MAC address. Computers communicate with each other using this MAC address, independently of the network protocol used. Every computer in a TCP/IP network uses the Address Resolution Protocol (ARP) to determine the MAC address of other computers.

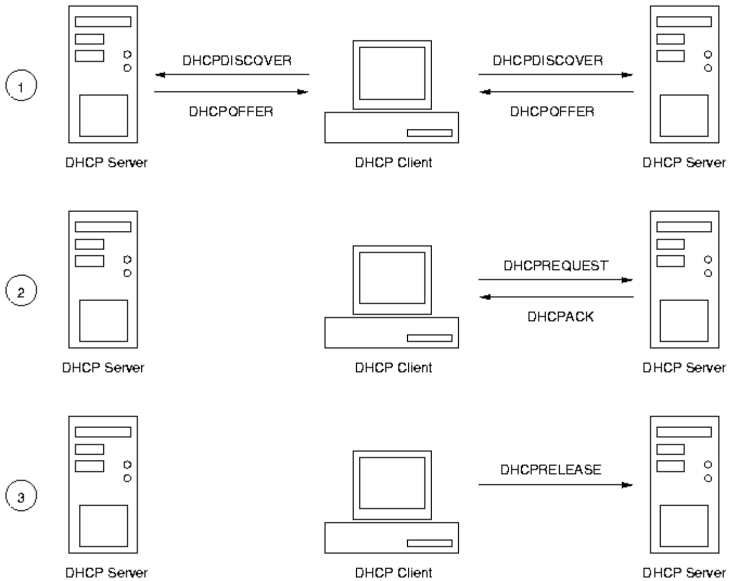
The information about the IP address of a computer located in the local subnet, together with the corresponding MAC address, is listed in the ARP table on the computer. This information is stored only in a cache; it's not written to a file.

If a machine sends data to another machine in the local network whose MAC address it does not yet know, it will subsequently send, or broadcast, a request for the MAC address to all the machines. The requested machine will then recognize, from the IP address, that it is being called and will replyto the machine that initiated the broadcast with its own MAC address.

DHCP functions in much the same way: A computer sends a broadcast containing its MAC address and expects that a DHCP server in the local network assigns an IP address for this MAC address. The DHCP server recognizes this machine based on the MAC address and assigns it an IP address.

This assignment procedure is outlined in the following figure.

Figure 2-1



The steps are described below:

1. On startup, the computer broadcasts a message to all the machines in the network, so it can find one or more DHCP servers (DHCPDISCOVER).

   If a DHCP server configured to respond to this request is on the network, it will answer the broadcasting machine's request by offering an IP address and possible additional information (DHCPOFFER).

2. The client chooses one of the responses it has received. One criteria for this selection is the time of response. The client then sends a request to the corresponding DHCP server for the IP address (DHCPREQUEST).

   The DHCP server, in turn, confirms this information (DHCPACK) and the client can now use the IP address.

When the validity period (the lease time) of the IP address expires, the client sends a request to the server again (DHCPREQUEST).

The server can either extend the duration of the IP address (DHCPACK) or prompt the client to request a new address. In this case, the server sends the notification DHCPNAK (negative acknowledgment).

3. If the client no longer needs the IP address, it sends a release notification to the DHCP server (DHCPRELEASE). Now the server can reassign the address.

A DHCP server can handle queries from DHCP clients as well as from BOOTP clients.

There are two ways to pass IP addresses with DHCP:

1. **Static assignment**. The administrator defines a fixed IP address for a machine. This IP address is assigned to the computer when it is booted. The allocation of the IP address host is based on the MAC address of the network card of the client.

   In this way, the client always receives the same address from the server.

2. **Dynamic assignment**. The DHCP server assigns the host an IP address from an address pool. The next time the machine is booted, the host will probably receive a different IP address. This simplifies the administration of a large network.

   The administrator defines the required parameters within the configuration of the DHCP server and, above all, determines from which range the dynamically assigned IP addresses should originate.

   After configuring the machines as DHCP clients, no additional configuration is necessary, even if extensive changes are made to the network structure, such as dividing it into new subnets.

Only the DHCP server must be configured accordingly, so that it can then disclose all the required information to the machines sending the requests.

Objective 2 Configure the DHCP Server

To configure a Linux machine as a DHCP server, the following packages must be installed:

- dhcp
- dhcp-server

The configuration is done using the files

- /etc/sysconfig/dhcpd
- /etc/dhcpd.conf

before the service is started by entering

/etc/init.d/dhcpd start

or

rcdhcpd start

To start the DHCP daemon automatically every time the machine is booted, create the required symbolic links by using **insserv**. When the DHCP server (dhcpd) starts, it reads the configuration parameters from the files /etc/sysconfig/dhcpd and /etc/dhcpd.conf.

These files contain information about the server itself as well as information about which clients should be assigned addresses and domains. If changes have been made to the configuration, dhcpd must be restarted (or at least reloaded).

To configure the DHCP server, you need to know the following:

- The Configuration File /etc/sysconfig/dhcpd
- The Configuration File /etc/dhcpd.conf
- The Log File
- Configure a DHCP Server with YaST

The Configuration File /etc/sysconfig/dhcpd

The file /etc/sysconfig/dhcpd contains configuration options which are submitted as parameters to the DHCP daemon when starting (see /etc/init.d/dhcpd). The first parameter defines the interface(s) on which the DHCP server listens for requests.

For example, if the DHCP server listens on the two interfaces eth0 and eth1, set the variable DHCPD_INTERFACE to

```
DHCPD_INTERFACE="eth0 eth1"
```

The DHCP server will listen only to the interfaces specified here.

Starting with kernel 2.6, the names assigned to the network interface are not guaranteed to be permanent. The interface called eth0 might be called eth1 after the next reboot. (The names can only change after reloading the network kernel module.) This can be quite annoying if the DHCP server is not listening on all interfaces.
You can rename interfaces in the file /etc/udev/rules.d/30-net_persistent_names.rules. For more detailed information, read /usr/share/doc/packages/sysconfig/README.Persistent_Interface_Names.

Two other variables enhance the security of the server:

```
DHCPD_RUN_CHROOTED="yes"
```

and

```
DHCPD_RUN_AS="dhcpd"
```

The first of these variables configures the DHCP server processes to run in a chroot environment. The new root directory for all processes is /var/lib/dhcp.

The second variable defines the user to be used for running the processes. Normally there is no reason to change the default settings of these variables.

The DHCP server can read additional configuration files that are included in the main configuration file. As the server processes are running in a chroot environment, these additional configuration files have to be copied into the chroot environment, too. The files will be copied automatically when the DHCP server is starting, if they are listed in /etc/sysconfig/dhcpd.

The following is an example:

```
DHCPD_CONF_INCLUDE_FILES="/etc/dhcpd.conf.shared /etc/dhcpd.conf.d”
```

As shown here, the name of a directory can also be provided. All files located in this directory will be included.

Use the option **DHCPD_OTHER_ARGS** if you want to start the dhcpd with some special arguments.

The arguments are described in the man page of dhcpd (man 8 dhcpd).

The Configuration File /etc/dhcpd.conf

The configuration file for the DHCP server is /etc/dhcpd.conf. Global definitions are made at the top of the configuration file. The parameters defined here apply to all subsequent sections unless they are explicitly overwritten in the respective sections. The entries in the configuration file belong to two categories:

- **Parameter statements**. They describe
  - How to do something (for example, define the time to which an IP address is assigned)

    - Whether to do something (for example, whether IP addresses should be assigned to unknown clients)
    - Which parameters should be provided to clients (for example, the IP address of the default gateway)
- **Declarations**. They describe the topology of the network, for example, describe the clients, or provide the address ranges from which to server clients.

Each statement have to be terminated using the semicolon (“;”).

In case of an error in the configuration file, dhcpd will not start but will print out an error message. This message can be used to locate the error in the configuration syntax.

To check the syntax of the configuration file without starting the DHCP server, use the start script with the parameter **syntax-check**:

```
da51:~ # rcdhcpd syntax-check
Checking syntax of /etc/dhcpd.conf:
Config is okay. Hope you also specified existent network devices ;)
Lease file is okay
da51:~ #
```

SUSE Linux Enterprise Server 10 ships with a sample configuration file for the DHCP server. You will not need all the configuration statements that are provided with this sample file. It is better to start with an empty configuration and to enter only those statements you really need.

Information on the configuration of the DHCP server is available at several locations:

- The man pages on your local system:
    - **man dhcpd** (DHCP server)
    - **man dhcpd.conf** (configuration file)
    - **man dhcp-options** (configuration options)
- In directories on your local system:

- /usr/share/doc/packages/dhcp/
- /usr/share/doc/packages/dhcp-server/

- On the web:
  - http://www.isc.org/products/DHCP/
  - http://www.dhcp-handbook.com
- In books:
  - Ralphs Droms and Ted Lemon, ***The DHCP Handbook*** (Sams Publishing)

We will explain how to configure DHCP in the following steps:

- A Simple Configuration
- Assign Fixed Addresses
- Group Host Declarations
- Configure DHCP Pools
- Use Classes

A Simple Configuration

Comments can be used at any location in the configuration file. They start with the hash sign ("#"). The rest of the line after the hash sign will be ignored.

The configuration file starts with the following lines:

```
#
# /etc/dhcpd.conf
#
ddns-update-style none;
```

Starting with DHCP server version 3, dynamic updates of a DNS server are possible. This means when the DHCP server assigns an IP address to a client, it can update the corresponding information on the DNS server. The statement describing how to do this dynamic update (ddns-update-style) is mandatory. If no dynamic update is done (as in this example), specify **none** as the parameter to this statement.

The following are statements on the lease times (the validity period for assigned IP addresses):

```
#
# specify default and maximum lease time
#
default-lease-time 86400;
max-lease-time 86400;
```

When a client requests an IP address without providing any information on the desired lease time, the IP address will be assigned for the specified default lease time (in this example 86400 seconds, which is one day).

At maximum you can enter a number of 2^{31}-1 seconds for the lease time. That is about 68 years.

Shortly before the assigned IP address expires, the client will request a renewal of the address. Normally, the lease time for this address will be extended.

Depending on its configuration, a client can request a specific lease time. Normally, this specific lease time request is accepted. We have to distinguish the two cases:

- If the requested lease time is shorter than the default lease time, the DHCP server will assign the IP address for the requested time.

- If the requested lease time is longer than the default lease time and if no maximum time has been specified the DHCP server will accept this. If the statement max-lease-time is present, this time will be the longest available.

In the example above, both times are the same. Setting a maximum lease time prohibits clients from requesting an infinite lease time (resulting in a permanent IP address).

In the next step, provide information on the DNS domain to be used:

```
#
# What is the DNS domain and where is the name server?
#
option domain-name "digitalairlines.com";
option domain-name-servers 10.0.0.254, 10.0.0.2;
```

These configuration options start with the keyword **option**.

If a list of name server addresses (separated by commas) is provided, the list reflects the order of preference for contacting a name server.

As the last parameter, specify the addresses of routers in the subnet:

```
#
# This is a router
#
option routers 10.0.0.254;
```

If several routers are specified here (separated by commas), the list reflects the order of preference for using these routers. The first router is the default gateway.

Finally, you need the declaration describing the range of addresses that can be used for assigning IP addresses to clients. This declaration starts with the keyword **subnet** and specifies the subnet and corresponding network mask:

```
#
# Which IP addresses may be assigned to the clients?
#
subnet 10.0.0.0 netmask 255.255.255.0
{
    range 10.0.0.101 10.0.0.120;
}
```

When a client requests an IP address, it will be assigned a free address from this range. Starting with version 3 of the DHCP server, assignment will start with the highest addresses (in this case, 10.0.0.120). If no parameters are defined inside this subnet declaration, all globally defined parameters will be used. There can be more than one range statement inside a subnet declaration.

The man pages for dhcp-options and dhcpd.conf provide more information on the available configuration options.

Assign Fixed Addresses

Using DHCP, you can assign a fixed IP address to a certain computer. Whenever this computer requests an IP address, it will always get the same address.

To assign a fixed address, the computer must be identified by its MAC address:

```
#
# Host da5 will always get the same address
#
host da5.digitalairlines.com
{
    hardware ethernet 00:90:27:51:4D:95;
    fixed-address 10.0.0.5;
}
```

To identify this host, the hardware statement is mandatory. Currently, only Ethernet and Token Ring are defined as hardware types.

If there are any specific settings required for this host (such as lease time or routers), they have to be provided inside the host statement.

If you are assigning fixed addresses to certain hosts, make sure that these addresses are outside all the defined address ranges.

Assigned fixed addresses are not logged in the log file that contains information about all the leases. Only dynamically assigned addresses are logged there.

Group Host Declarations

Instead of providing specific settings for several hosts in individual host statements, the hosts can be grouped in one declaration. The declaration starts with the **group** statement and contains a list of the hosts and the corresponding settings. The following is an example:

```
#
# This group of hosts needs specific settings
#
group {
    default-lease-time 43200;
    max-lease-time 86400;
    option routers 10.0.0.250;
    host da6.digitalairlines.com
    {
        hardware ethernet 00:90:27:51:4C:95;
    }
    host da7.digitalairlines.com
    {
        hardware ethernet 00:90:27:51:4C:96;
    }
    host da8.digitalairlines.com
    {
        hardware ethernet 00:90:27:51:4C:97;
    }
}
```

All other settings will be used as defined in the global settings.

Configure DHCP Pools

The **pool** declaration can be used to specify a pool of addresses that will be treated differently than any other pool of addresses, even on the same network segment or subnet.

For example, you may want to provide a large set of addresses that can be assigned to DHCP clients that are registered to your DHCP server, while providing a smaller set of addresses, possibly with short lease times, that are available for unknown clients.

To do this, you would set up a pair of **pool** declarations:

```
subnet 10.0.0.0 netmask 255.255.255.0 {
    option routers 10.0.0.254;

# Unknown clients get this pool.
    pool {
        option domain-name-servers 10.0.0.254;
        max-lease-time 300;
        range 10.0.0.100 10.0.0.250;
        allow unknown-clients;
    }

# Known clients get this pool.
    pool {
        option domain-name-servers 10.0.0.251, 10.0.0.252;
        max-lease-time 28800;
        range 10.0.0.5 10.0.0.99;
        deny unknown-clients;
    }
}
```

It is also possible to set up entirely different subnets for known and unknown clients. “Known clients” mean, that an IP address is assigned to a MAC address using the **host** statement. Pools exist at the level of shared networks, so address ranges within pool declarations can be on different subnets.

As you can see in the preceding example, pools can have permit lists that control which clients are allowed access to the pool and which aren't.

Each entry in a pool's permit list is introduced with the allow or deny keyword. If a pool has a permit list, then only those clients that match specific entries on the permit list will be eligible to be assigned addresses from the pool.

If a pool has a deny list, then only those clients that do not match any entries on the deny list will be eligible. If both permit and deny lists exist for a pool, then only clients that match the permit list and do not match the deny list will be allowed access.

Use Classes

Clients can be separated into classes and are then treated differently depending on what class they are in. This can be used, for example, to prohibit the DHCP server from offering IP addresses to computers (those would have to get their information from another DHCP server).

The definition of a class always starts with the **class** statement, followed by the name of the class (you can choose any name here). After this definition, the parameters are needed how to identify the members of this class. A simple example looks like this:

```
class "unwanted-clients"
{
    match if substring(hardware, 1, 3) = 00:11:22;
    deny booting;
}
```

The **match if** statement in the example specifies that the members of this class should be identified by their hardware (MAC) address. This line identifies all MAC addresses where the first three bytes match "00:11:22". All clients matching this definition are grouped into this class automatically. The **deny booting** statement in the next line defines that the DHCP server should not provide any IP addresses to these clients.

A more complex example makes use of the so-called pools into which you can separate the IP addresses to be used in the network:

```
class "apple-macs"
{
    match if substring(hardware, 1, 3) = 01:23:45;
}

subnet 10.0.0.0 netmask 255.255.255.0
{
    ...
    pool {
        allow members of "apple-macs";
        range 10.0.0.101 10.0.0.150;
    }
    pool {
        deny members of "apple-macs";
        range 10.0.0.201 10.0.0.250;
    }
}
```

This configuration defines a class apple-macs that contains all clients which match the given MAC address (the first three bytes are 01:23:45). The members of this class will be assigned IP addresses from the range 10.0.0.101 to 10.0.0.150, while all clients which are not members of this class will be assigned IP addresses from the range 10.0.0.201 to 10.0.0.250.

The Log File

The IP addresses assigned by the DHCP server are logged to the file **/var/lib/dhcp/db/dhcpd.leases**. For each address assigned to a client, several pieces of information are recorded, such as

- When the address was assigned
- When the lease time will end
- What the MAC address of the client is
- The host name of the client (if submitted during the request)

Whenever a new address is requested by a client, the corresponding entry is added to the end of the log file. When dhcpd is started, the log file is read so that the DHCP server knows about currently assigned IP addresses.

This file is required before you start the DHCP server for the first time. If the file was not created automatically during DHCP installation, it has to be created manually. After starting dhcpd, a copy of the previous version of the log file will be created using the name

/var/lib/dhcp/db/dhcpd.leases~

To prevent a log file from becoming too large, by adding more and more entries it will be re-created occasionally.

In the following example, one machine has been assigned the IP address 10.0.0.120, and another one has been assigned the IP address 10.0.0.119. The first address has been assigned for one hour, the second for twelve hours.

```
lease 10.0.0.120
{
    starts 1 2006/06/14 08:06:51;
    ends 1 2006/06/14 09:06:51;
    binding state active;
    next binding state free;
    hardware ethernet 00:50:ba:be:33:1c;
    uid "\001\000P\272\2763\034";
    client-hostname "da5";
}

lease 10.0.0.119
{
    starts 1 2006/06/14 08:08:06;
    ends 1 2006/06/14 20:08:06;
    binding state active;
    next binding state free;
    hardware ethernet 08:00:46:b6:d8:ce;
    uid "\001\010\000F\266\330\316";
    client-hostname "da6";
}
```

The following explains the entries:

- The keyword **starts** is followed by the time the address was assigned. The keyword **ends** is followed by the time the address expires (the time at which the address is released).

  The first number following the corresponding keyword indicates the day of the week (0 stands for Sunday, 6 for Saturday). Next, the date and clock time follow (always as Greenwich Mean Time, GMT).

- The keywords **binding state** and **next binding state** provide information on how the address can be assigned. If the DHCP server is not configured for failover, the only possible parameters are **active** and **free**. In a failover setup, you can use additional parameters like **backup**.
- The keyword **hardware** specifies the MAC address of the network card. Addresses that are assigned permanently to specific clients via the entry **hardware** in the file /etc/dhcpd.conf are not listed in this log file.
- The keyword **uid** is followed by a client identification (provided the client transferred this information when the query was made).

Most clients send their host names when requesting an IP address via DHCP. If this is the case, the host name appears after the keyword **client-hostname** in the lease file.

Configure a DHCP Server with YaST

The DHCP server can be configured with YaST by selecting

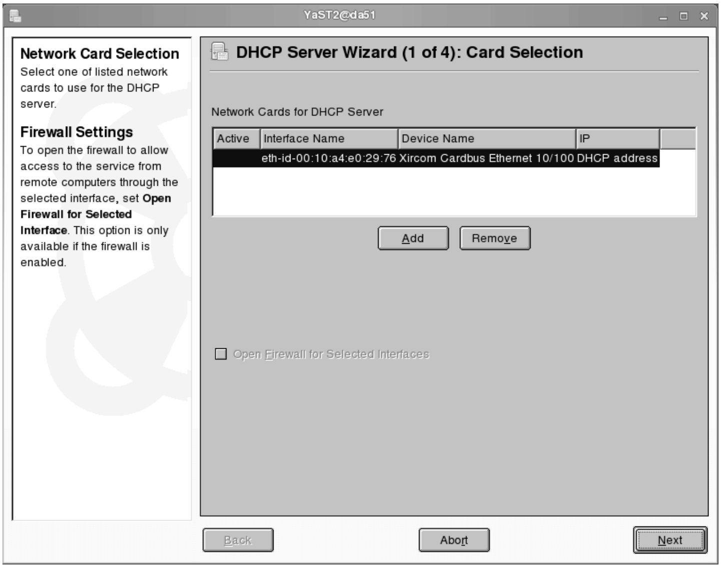
YaST > Network Services > DHCP Server

When this YaST module is started for the first time, a configuration wizard is started, enabling a simple, basic configuration in four steps.

Do the following:

1. Select the network interface on which the DHCP server is to listen:

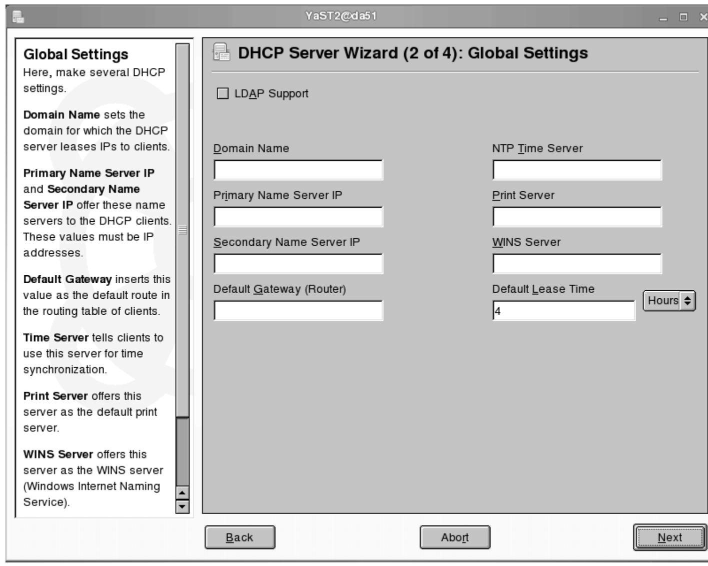
Figure 2-2



To do this, select the interface in the upper list and select **Add**. A small "x" appears in the Active column.

2. Set global settings such as the domain name and the IP addresses of name servers:

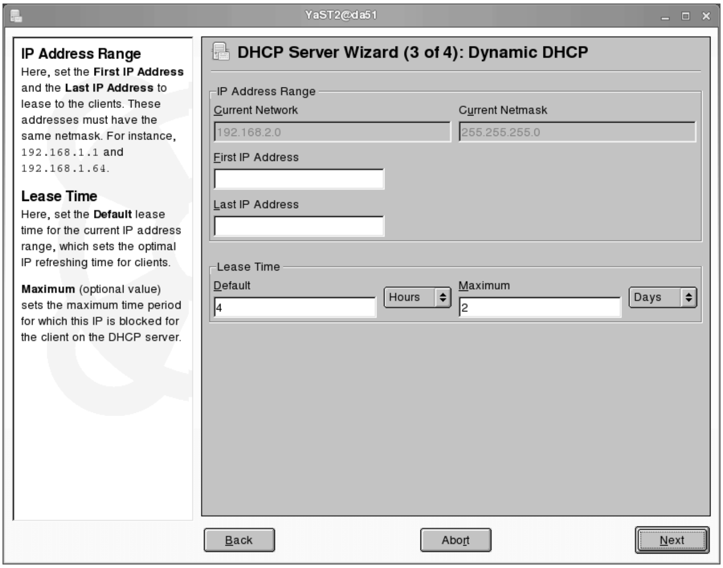
Figure 2-3



In SUSE Linux Enterprise Server 10, the data of the DHCP server can be managed in an LDAP directory. To enable this function, **LDAP Support** must be activated.

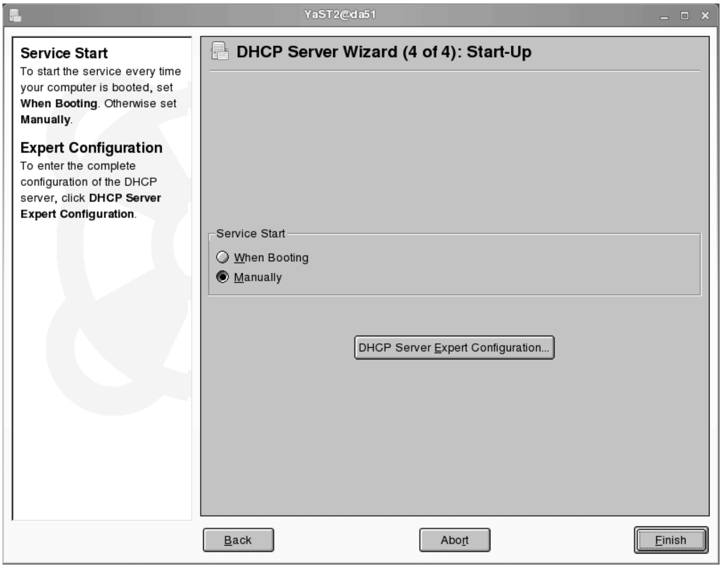
3. Set the address space from which the addresses are assigned dynamically and specity the lease time; then select **Next**:

Figure 2-4



4. In the last step you have to specify when to start the DHCP server.

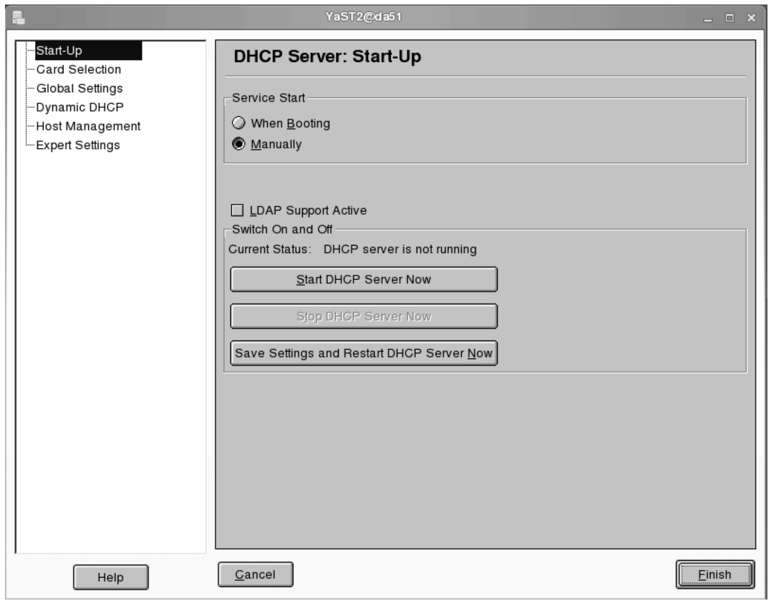
Figure 2-5



This wizard for configuring the DHCP server is only started when the DHCP server module is used for the first time.

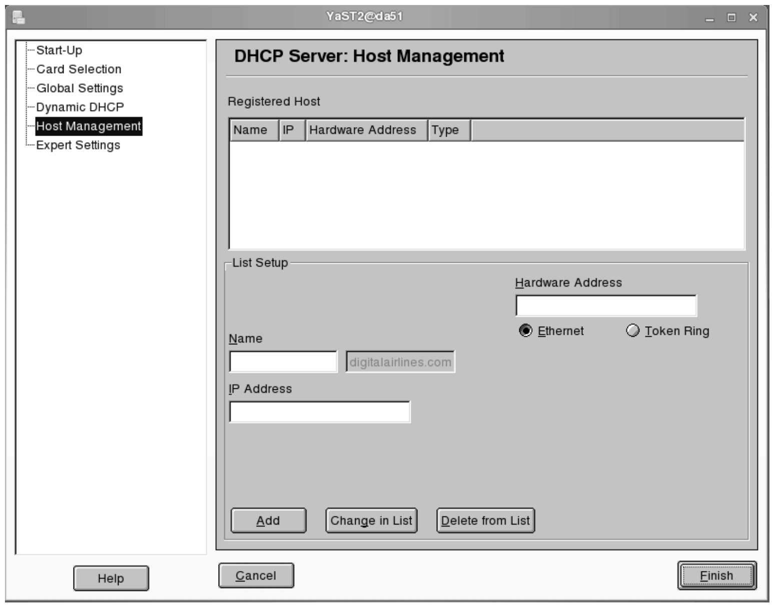
After this, the module looks different, but you can modify all settings made with the wizard:

Figure 2-6



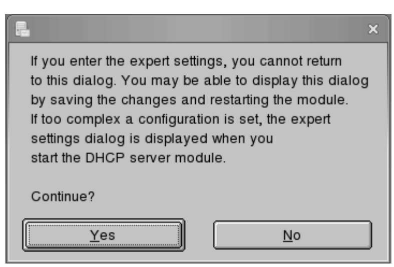
You can enter the hosts that are to be assigned static IP addresses based on their MAC addresses under **Host Management**.

Figure 2-7



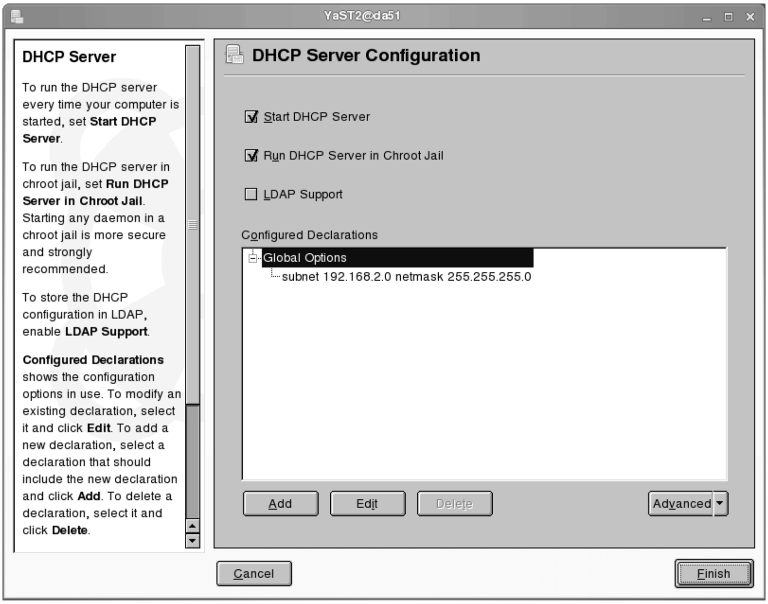
To perform further settings, such as the dynamic update of the name server by the DHCP server, select **Expert Settings**. A warning appears that you are not able to return to the DHCP server management from the Expert Settings dialog.

Figure 2-8



If you select **Yes**, the settings are modified; the next time the YaST module is launched, the expert module will be started instead of the standard module.

Figure 2-9



One important aspect of the expert module is
Run DHCP Server in Chroot Jail

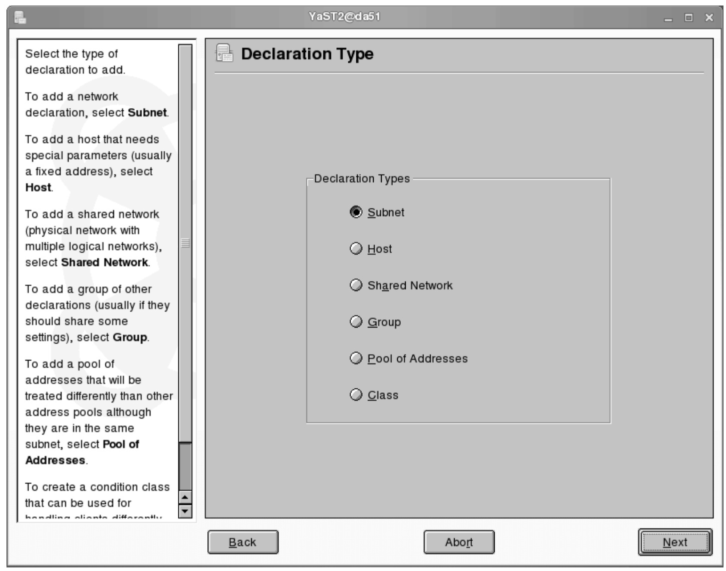
By default, the DHCP server in SUSE Linux Enterprise Server runs in a chroot environment. The daemon is started by the user nobody. Upon startup, all relevant configuration files are copied to the directory /var/lib/dhcp/, which serves as the root directory of the daemon. Disable this option if you do not want to use this security option.

In this way, the variable DHCPD_RUN_CHROOTED in the file /etc/sysconfig/dhcpd is set to **no**.

The configured declarations are displayed in the main field. You have the following options:

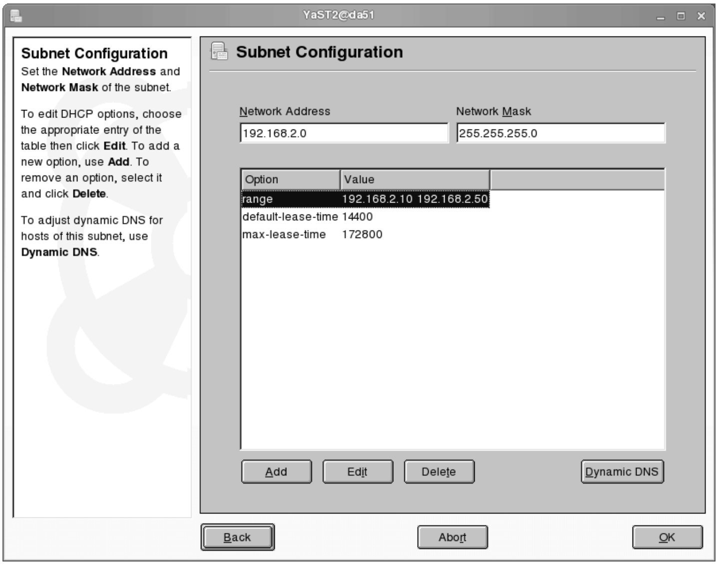
- Select **Add** to add further declarations, including addresses to be assigned statically to specific hosts.

Figure 2-10



- Select **Edit** and **Dynamic DNS** to configure the dynamic modification of zone files of the name server for subnet declarations.

Figure 2-11



- Select **Advanced** to see the server protocols (entries of dhcpd in /var/log/messages) and to manage the authentication keys for dynamic DNS updates.

Exercise 2-1 Configure the DHCP Server

In this exercise, you install and configure a DHCP server for the domain digitalairlines.com.

You will find this exercise in the workbook.

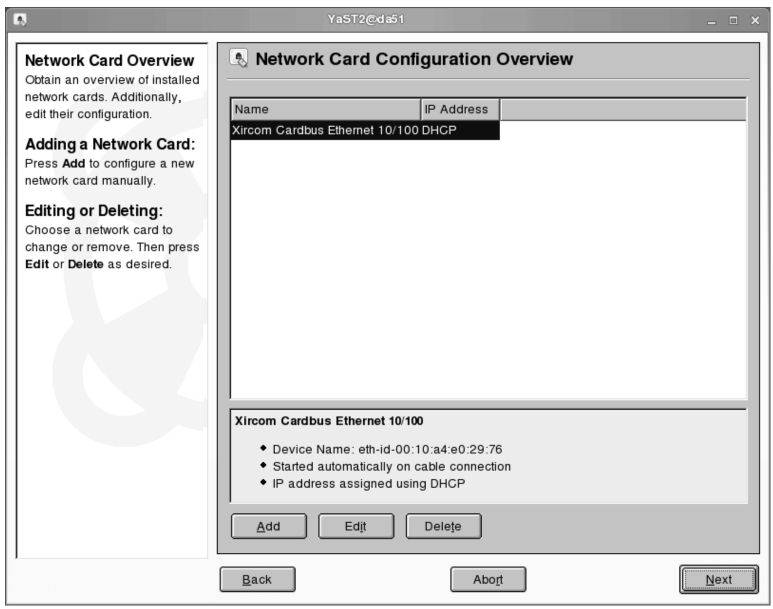
(End of Exercise)

Objective 3 Configure DHCP Clients

To run a Linux host as a DHCP client, a corresponding daemon must be installed on the machine. dhcpcd (*DHCP client daemon*) is included in the standard installation. To configure the DHCP client, the host must be instructed to obtain its IP address from a DHCP server.

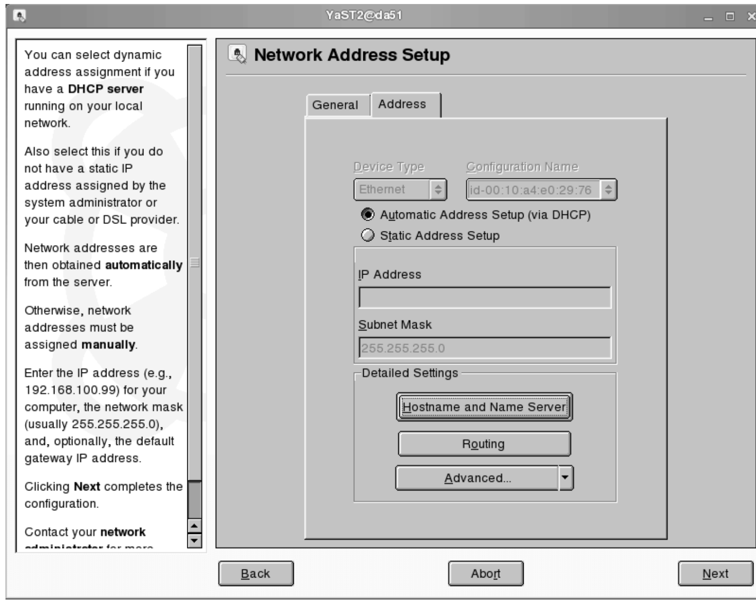
You can configure the DHCP client manually or using YaST. Select **Network Devices > Network Card** and **Traditional Method with ifup**.

Figure 2-12



From the list, select the network card that should be configured by DHCP and select **Edit**.

Figure 2-13



In this dialog, specify that the IP address and subnet mask should be requested from the DHCP server when the computer is booted by activating the option **Automatic Address Setup (via DHCP)**.

You can also use DHCP to allocate host names. To achieve this, select **Change Host Name Via DHCP**, which is available under **Host Name and Name Server**. When DHCP assigns an IP address, it contacts the DNS server to find out the corresponding name. This name is used to update the client.

Alternatively, the client can transmit its name to the DHCP server that sends it to the DNS server if dynamic DNS via DHCP is enabled. In this case, the clients must have been assigned local host names to prevent various DHCP clients from being assigned the same name in DNS.

Even when a system is newly installed on a computer, DHCP can be activated directly in the network configuration.

The configuration files for the network interfaces are located in the /etc/sysconfig/network/ directory. Each interface has its own configuration file; for example, /etc/sysconfig/network/ifcfg-eth-id-00:0c:29:ed:c0:03. This file contains the values with which a network card should be activated. If an interface should be configured with DHCP, the entry might look like the following:

```
BOOTPROTO='dhcp'
BROADCAST=''
ETHTOOL_OPTIONS=''
IFPLUGD_PRIORITY='20'
IPADDR=''
MTU=''
NAME='Xircom Cardbus Ethernet 10/100'
NETMASK='255.255.255.0'
NETWORK=''
REMOTE_IPADDR=''
STARTMODE='ifplugd'
UNIQUE='rBUF.Iew_4A28T26'
USERCONTROL='no'
_nm_name='bus-pci-0000:07:00.0'
```

The following explains the three most important entries:

- **BOOTPROTO=”dhcp”** and **STARTMODE=”ifplugd”** ensure that the DHCP client is launched automatically during the boot process. Otherwise, **BOOTPROTO** is set to **static**.
- **UNIQUE=”rBUF.Iew_4A28T26”** is generated by YaST, if the configuration was written by it.

Options for the DHCP client are stored in the file /etc/sysconfig/network/dhcp.

The following are some of the important options:

- **DHCLIENT_BIN**. Specifies which DHCP client should be used. Possible values are **dhcpd** (DHCP client daemon) and **dhclient** (ISC dhclient)

  By default, it is set to **dhcpd**.
- **DHCLIENT_DEBUG**. If set to **yes**, any debug output will be written to /var/log/messages.

  By default, it is set to **no**.
- **DHCLIENT_SET_HOSTNAME**. Should the DHCP client set the hostname? By default, it is set to **yes**.
- **DHCLIENT_MODIFY_RESOLV_CONF**. Should the file /etc/resolv.conf be modified by the client? By default, it is set to **yes**.
- **DHCLIENT_SET_DEFAULT_ROUTE**. Should the DHCP client set a default route (default Gateway)? By default, it is set to **yes**.
- **DHCLIENT_MODIFY_NTP_CONF**. If set to **yes**, the file /etc/ntp.conf is modified as needed if the DHCP server is enabled to send NTP server information (**option ntp-servers**) and if xntpd is used.

  By default, it is set to **no**.

- **DHCLIENT_SET_DOMAINNAME**. Should the DHCP client set the domain name? By default, it is set to **yes**.
- **DHCLIENT_KEEP_SEARCHLIST**. This option affects the contents of the file /etc/resolv.conf.

  It defines whether the DHCP client should combine a new name server search list, as received from the DHCP server, with the existing one (**yes**).

  The default is **yes**.
- **DHCLIENT_HOSTNAME_OPTION**. Specifies a string used for the hostname option field when dhcpcd sends DHCP messages. Some DHCP servers will update nameserver entries (dynamic DNS).

  By default the current hostname is sent (**AUTO**), if one is defined in /etc/HOSTNAME. Use **DHCLIENT_HOSTNAME_OPTION** to override the default hostname with another hostname, or leave empty to not send a hostname.
- **DHCLIENT_CLIENT_ID**. Use this option to set a client ID.

  By default, this ID is equivalent to the MAC address of the client network interface.
- **DHCLIENT_SCRIPT_EXE**. This option allows you to specify a script to run when dhcpcd configures or disables the interface.

  A sample script is in /usr/share/doc/packages/dhcpcd/dhcpcd.exe.

  For instance, scripts that are supposed to be run by dhcpcd (such as dhcpdcd.exe) can be placed in /etc/sysconfig/network/scripts/.
- **DHCLIENT_MODIFY_SMB_CONF**. Should the DHCP client modify /etc/samba/dhcp.conf? By default, it is set to **yes**.

The data for a configured interface as received by the DHCP client daemon dhcpcd from the DHCP server are stored in a file with a filename similar to /var/lib/dhcpcd/dhcpcd-eth0.cache (in this case, for the interface eth0). This data is also available in human-readable format, in the file /var/lib/dhcpcd/dhcpcd-eth0.info. Whenever the DHCP client is restarted, it will try to retrieve the same IP address as before—the one stored in the file.

If you enter the command **ifstatus-dhcp eth0** on a DHCP client, information is read from the file and displayed.

Exercise 2-2 Configure DHCP Clients

In this exercise, you activate the DHCP client using YaST.

You will find this exercise in the workbook.

(End of Exercise)

Objective 4 Configure a DHCP Relay Server

As routers do not route packets addressed to the broadcast address of the network, the DHCP server must be accessible for the DHCP clients in the local network.

Instead of implementing a DHCP server in every local network, a DHCP relay server can be installed on the routers for the purpose of receiving the client broadcasts via one interface and forwarding the broadcasts to the central server via the other interface.

The package is **dhcp-relay**. The configuration of the DHCP relay is located in the file /etc/sysconfig/dhcrelay. The network interfaces are entered under DHCRELAY_INTERFACES; for example **DHCRELAY_INTERFACES=”eth0 eth1”**

The IP address of the DHCP server to which the relay server will forward the client queries must be specified under **DHCRELAY_SERVERS**.

On the DHCP server itself, a suitable subnet declaration is required for the clients in the network behind the router.

However, the entries do not differ from the entries required for clients in the server’s local network.

Objective 5 Use DHCP and Dynamic DNS

If you want clients that are assigned variable IP addresses via DHCP to be accessible under their respective names, the DNS server must be informed about the host name and the changed IP addresses.

This can be done in two ways:

- The client transmits its name to the DHCP server and gets an IP address from the DHCP server. The DHCP server transmits the host name and the IP address to the DNS server.
- The client gets its name and IP address from the DHCP server. The DHCP server transmits the host name and IP address to the DNS server.

To enable interaction between the DHCP server and DNS server, you must modify the respective configuration files (/etc/named.conf and /etc/dhcpd.conf).

Because the DNS server can only accept changes from the authorized DHCP server, the updates must be protected cryptographically using a key.

This key is generated using the program **dnssec-keygen**.

The following parameters must be specified:

- **-a**. The algorithm (**RSAMD5**, **RSASHA1**, **DSA**, **DH** or **HMAC-MD5**)
- **-b**. The key length in bits (the key length depends on the used algorithm).
- **-n**. The name type, for example:
  - **HOST.** For a host-specific key
  - **USER.** For a user-specific key
  - **ZONE.** For the protected transmission of zone files

- ***name of the key*** Used in the configuration files

The following is an example:

```
da51:~ # dnssec-keygen -a HMAC-MD5 -b 128 -n HOST dhcp-dns
```

Two files are generated in the current directory:

- Kdhcp-dns.+157+23165.key
- Kdhcp-dns.+157+23165.private

(The second number is a random number).

The name of the files is written on the screen by the dnssec-keygen command.

```
da51:~ # dnssec-keygen -a HMAC-MD5 -b 128 -n HOST dhcp-dns
Kdhcp-dns.+157+23165
```

These files contain the key in clear text:

```
da51:~ # cat Kdhcp-dns.+157+23165.key
dhcp-dns. IN KEY 512 3 157 UsiG5qub101MHk0jzoKQgw==
```

Enter this key and information about the affected zone in the configuration files.

In the file /etc/named.conf, append the following after **options**:

```
key dhcp-dns {
    algorithm HMAC-MD5;
    secret UsiG5qub101MHk0jzoKQgw==;
};

zone "digitalairlines.com" in {
    type master;
    file "digitalairlines.zone";
    allow-update { key dhcp-dns ;};
};

zone "10.0.0.zone" in {
    type master;
    file "digitalairlines.arpa";
    allow-update { key dhcp-dns ;};
};
```

Because this file contains the key in clear text, the permissions of the configuration file must be set so that the file is not readable for all users (this is the default for SUSE Linux Enterprise Server 10):

```
da51:~ # ls -l /etc/named.conf
da51:~ # -rw-r----- 1 root named 3846 Jun 30 2006 /etc/named.conf
```

If you include the key using the include statement, it is not necessary to change the file permissions.

On the DHCP server, enter the following in the file /etc/dhcpd.conf:

```
ddns-update-style interim;
ddns-updates on;

key dhcp-dns {
    algorithm HMAC-MD5;
    secret UsiG5qub101MHk0jzoKQgw==;
}

# The dot after 0.0.10.in-addr.arpa. is essential!
zone 0.0.10.in-addr.arpa. {
    key dhcp-dns;
}

# The dot after digitalairlines.com is essential!
zone digitalairlines.com. {
    key  dhcp-dns;
}
```

The two important parameters are described below:

- **ddns-update-style**. Describes how the DNS is updated. A DNS server usually maintains two resource records for each client. One maps FQDNs to IP addresses using A records. The other maps the IP address to the FQDN using PTR records (Pointer). .

  Two update-styles are common:

  - **ad-hoc**. The A record will contain the IP address that the client was assigned in its lease. If there is already an A record with the same name in the DNS server, no update of either the A or PTR records will occur.

    If the A record update succeeds, a PTR record update for the assigned IP address will be done, pointing to the A record. This update is unconditional - it will be done even if another PTR record of the same name exists.

When the client's lease expires, the DHCP server will remove the client's A and PTR records from the DNS database. If the client releases its lease by sending a DHCPRELEASE message, the server will likewise remove the A and PTR records.

- **interim**. Unlike the ad-hoc style, the DHCP server does not necessarily always update both the A and the PTR records. The FQDN option includes a flag which, when sent by the client, indicates that the client wants to update its own A record. In that case, the server can be configured either to honor the client's intentions or ignore them.

  If the server is configured to allow client updates, and the client sends a fully qualified domain name, the server will use that name the client sent to update the PTR record. If the client also indicates that it wants to update its own A record, the DHCP server will not attempt to set up an A record for the client, but does set up a PTR record for the IP address that it assigns the client, pointing at the FQDN supplied by the client.

  If the server is configured not to allow client updates or if the client doesn't want to do its own update, the server will simply choose a name for the client, possibly using the hostname supplied by the client. If the client sends an FQDN, the server uses only the leftmost part of it. However, if the server configuration contains an option **ddns-hostname**, this one takes precedence over a possible client supplied FQDN or hostname.

The ad-hoc style is deprecated.

- **ddns-updates**. Activates dynamic DNS.

As this file contains the key in clear text, you must modify the file permissions of the configuration file to prevent it from being readable for all users:

```
da51:~ # chmod 600 /etc/dhcpd.conf
```

The client-specific entries depend on whether the client transmits its host name to the DHCP server or gets its host name from the DHCP server.

The following happens:

- The Client Transmits Its Name to the DHCP Server
- The DHCP Server Assigns a Name to the Client

Then the zone files are updated.

The Client Transmits Its Name to the DHCP Server

In case the DHCP server gets the host name from the client, the configuration within a **subnet** declaration is the same as the configuration without dynamic DNS.

```
subnet 10.0.0.0 netmask 255.255.255.0 {
    range 10.0.0.50 10.0.0.90;
    # options may also be put here if they are not global
}
```

The DHCP Server Assigns a Name to the Client

In this case, a client must always be assigned the same host name, even if the IP address changes.

The allocation of the name to the host is based on the MAC address of the network card.

An entry for a client could look as follows (the IP address assigned to the client is derived from a **subnet** declaration that is independent from the **host** entry):

```
host da5.digitalairlines.com {
  # The following entries concern the host
  # with this MAC address:
  hardware ethernet 00:90:27:a4:96:1f;

  # da5 and digitalairlines.com are transmitted to the DNS server:
  ddns-hostname "da5";
  ddns-domainname "digitalairlines.com";

  # The host name da5 is transmitted to the client:
  option host-name "da5";
}
```

The client has to be configured in such a way that it accepts the host name via DHCP.

The easiest way to do this is to use YaST. Do the following:

1. Select **YaST2 > Network Devices > Network Card**.
2. Select a network card.
3. Under **Host Name and Name Server**, select **Change Host Name via DHCP**.

   YaST changes the entry DHCLIENT_SET_HOSTNAME in /etc/sysconfig/network/dhcp to **yes**.

The host name entered in the file /etc/dhcpd.conf under **option host-name** will be displayed at the login prompt.

The log file (/var/log/messages) of the DNS server and the DHCP server provide information about the address assignment and the success or failure of the DNS server update.

Exercise 2-3 Use DHCP and Dynamic DNS

In this exercise, you configure dynamic DNS for your DHCP server.

You will find this exercise in the workbook.

(End of Exercise)

Objective 6 Configure DHCP Failover

This objective covers the following topics:

- Basics of DHCP Failover
- Configure Failover

Basics of DHCP Failover

The failover protocol allows two DHCP servers (and no more than two) to share a common address pool. Each server will have about half of the available IP addresses in the pool at any given time for allocation.

If one server fails, the other server will continue to renew leases out of the pool, and will allocate new addresses out of the half of available addresses that it had before communication with the other server was lost.

When a server starts that has not previously communicated with its failover peer, it must establish communications and synchronize with the peer before it can serve clients. This can happen either because you have just configured your DHCP servers to perform failover for the first time, or because one of your failover servers has failed and lost its database.

The initial recovery process is designed to ensure that when one failover peer loses its database and then resynchronizes, any leases that the failed server gave out before it failed will be honored. When the failed server starts up, it notices that it has no saved failover state and attempts to contact its peer.

When it has established contact, it asks the peer for a complete copy of its lease database. The peer then sends its complete database and sends a message indicating that it is done. The failed server then waits until MCLT (*Maximum Client Lead Time*) has passed. Once MCLT has passed, both servers make the transition back into normal operation.

This waiting period ensures that any leases the failed server may have given out while out of contact with its partner will have expired.

While the failed server is recovering, its partner remains in the partner-down state, which means that it is serving all clients. The failed server provides no service at all to DHCP clients until it has made the transition into normal operation.

In the case where both servers detect that they have never before communicated with their partner, they both come up in this recovery state and follow the procedure we have just described. In this case, no service will be provided to DHCP clients until MCLT has expired.

Configure Failover

In a failover configuration you have to decide which server acts as the primary and which acts as the secondary server. Both servers need to have the same configuration for the basic DHCP service they are sharing. They only differ in the failover setup. Therefore, it is recommended to have a common configuration file on both machines which is included in the file /etc/dhcpd.conf.

The configuration file for the primary server looks like this:

```
failover peer "digitalairlines" {
        primary;
        address 10.0.0.10;
        port 847;
        peer address 10.0.0.12;
        peer port 647;
        max-response-delay 180;
        mclt 1800;
        split 128;
        load balance max seconds 3;
}

# Now we include the identical configuration on both
# machines
include "/etc/dhcpd.conf.master";
```

The statements in this example peer declaration are as follows:

- **primary** and **secondary**. Determines whether the server is primary or secondary, as described earlier.
- **address**. Specifies the IP address or DNS name on which the server should listen for connections from its failover peer.
- **port**. Specifies the TCP port on which the server should listen for connections from its failover peer (default: 647 and 847). This parameter must be specified, because the failover protocol does not yet have a reserved TCP port number.
- **peer address**. Specifies the IP address or DNS name to which the server should connect to reach its failover peer.
- **peer port**. Specifies the TCP port to which the server should connect to reach its failover peer for failover messages. This parameter must be specified, because the failover protocol does not yet have a reserved TCP port number. The **peer port** can be the same as the **port**.
- **max-response-delay**. Specifies how many seconds the DHCP server waits without receiving a message from its failover peer before it assumes that connection has failed.

This number should be small enough that a transient network failure breaks the connection will not result in the servers being out of communication for a long time, but large enough that the server isn't constantly making and breaking connections.

This parameter must be specified.

- **mclt**. Defines the *Maximum Client Lead Time*. It must be specified on the primary, and can be specified also on the secondary server. This is the length of time for which a lease may be renewed by either failover peer without contacting the other.

  The longer you set this, the longer it will take for the running server to recover IP addresses after moving into PARTNER-DOWN state. The shorter you set it, the more load your servers will experience when they are not communicating.

  A value of 1800 is recommended.

- **split**. Specifies the split between the primary and secondary server for the purposes of load balancing.

  Whenever a client makes a DHCP request, the DHCP server runs a hash on the client identification. If the hash comes out to less than the split value, the primary answers. If it comes out to equal to or more than the split, the secondary answers.

  A meaningful value is 128 and can only be configured on the primary server.

- **load balance max seconds**. Specifies a cutoff after which load balancing is disabled. The cutoff is based on the number of seconds since the client sent its first DHCPDISCOVER or DHCPREQUEST message.

  The man pages recommend setting this to 3 or 5. The effect of this is that if one of the failover peers gets into a state where it is responding to failover messages but not responding to some client requests, the other failover peer will take over its client load automatically as the clients retry.

The configuration file for the secondary server looks like this:

```
failover peer "digitalairlines” {
        secondary;
        address 10.0.0.12;
        port 647;
        peer address 10.0.0.10;
        peer port 847;
        max-response-delay 180;
        load balance max seconds 3;
}

# Now we include the identical configuration on both
# machines
include "/etc/dhcpd.conf.master";
```

The differences to the primary server configuration are the statement “secondary”, the missing statements **mclt** and **split** and the interchanged values of **address**, **port**, **peer address** and **peer port**.

The main DHCP server configuration is contained in the file /etc/dhcpd.conf.master, which included in both configurations.

In order to find this file, it needs to be copied into the chroot environment of the DHCP server. The best way to achieve this is to modify the variable DHCPD_CONF_INCLUDE_FILES in /etc/sysconfig/dhcpd:
DHCPD_CONF_INCLUDE_FILES="/etc/dhcpd.conf.master"

The master configuration file need to be modified:

```
ddns-update-style none;

default-lease-time 86400;
max-lease-time 86400;

option domain-name "digitalairlines.com";
option domain-name-servers 10.0.0.254;

option routers 10.0.0.254;

subnet 10.0.0.0 netmask 255.255.255.0 {
        pool {
                failover peer "digitalairlines";
                deny dynamic bootp clients;
                range 10.0.0.101 10.0.0.120;
        }
}
```

All failover configurations have to be defined in a **pool** statement. If you use several pools, you need to define the failover configuration in each of them.

In order to be aware of the name of the failover configuration, it has to be defined before the pool definition. That is why the include statement is written at the end of /etc/dhcpd.conf.

Failover is not supported on address allocation pools that contain addresses allocated to bootp clients. Therefore, the statement
deny dynamic bootp clients;
has to be defined.

When starting the DHCP server on the primary, you will see messages like these in /var/log/messages:

```
Jul  4 11:52:29 da10 dhcpd: failover peer digitalairlines:
I move from recover to startup
Jul  4 11:52:44 da10 dhcpd: failover peer digitalairlines:
I move from startup to recover
Jul  4 11:54:57 da10 dhcpd: failover peer digitalairlines:
peer moves from unknown-state to recover
Jul  4 11:54:57 da10 dhcpd: failover peer digitalairlines:
requesting full update from peer
Jul  4 11:54:57 da10 dhcpd: Sent update request all message
to digitalairlines
Jul  4 11:54:57 da10 dhcpd: failover peer digitalairlines:
peer moves from recover to recover
Jul  4 11:54:57 da10 dhcpd: failover peer digitalairlines:
requesting full update from peer
Jul  4 11:54:57 da10 dhcpd: Sent update done message to
digitalairlines
Jul  4 11:54:57 da10 dhcpd: Update request all from
digitalairlines: nothing pending
Jul  4 11:54:57 da10 dhcpd: failover peer digitalairlines:
peer update completed.
Jul  4 11:54:57 da10 dhcpd: failover peer digitalairlines:
I move from recover to recover-done
Jul  4 11:54:57 da10 dhcpd: failover peer digitalairlines:
I move from recover-done to normal
Jul  4 11:54:57 da10 dhcpd: failover peer digitalairlines:
peer moves from recover-done to normal
Jul  4 11:54:57 da10 dhcpd: pool 800e4138 10.0.0/24 total
20  free 20  backup 0  lts -10
Jul  4 11:54:57 da10 dhcpd: pool 800e4138 10.0.0/24  total
20  free 20  backup 0  lts 10
```


On the secondary server (which is started later), messages like these will appear in /var/log/messages:

```
Jul  4 11:55:52 da12 dhcpd: failover peer digitalairlines:
I move from recover to startup
Jul  4 11:55:52 da12 dhcpd: failover peer digitalairlines:
peer moves from unknown-state to recover
Jul  4 11:55:52 da12 dhcpd: failover peer digitalairlines:
requesting full update from peer
Jul  4 11:55:52 da12 dhcpd: failover peer digitalairlines:
I move from startup to recover
Jul  4 11:55:52 da12 dhcpd: Sent update request all message
to digitalairlines
Jul  4 11:55:52 da12 dhcpd: Sent update done message to
digitalairlines
Jul  4 11:55:52 da12 dhcpd: Update request all from
digitalairlines: nothing pending
Jul  4 11:55:52 da12 dhcpd: failover peer digitalairlines:
peer update completed.
Jul  4 11:55:52 da12 dhcpd: failover peer digitalairlines:
I move from recover to recover-done
Jul  4 11:55:52 da12 dhcpd: failover peer digitalairlines:
peer moves from recover to recover-done
Jul  4 11:55:52 da12 dhcpd: failover peer digitalairlines:
I move from recover-done to normal
Jul  4 11:55:52 da12 dhcpd: failover peer digitalairlines:
peer moves from recover-done to normal
Jul  4 11:55:52 da12 dhcpd: pool 800e40f8 10.0.0/24 total
20  free 20  backup 0  lts 10
Jul  4 11:55:52 da12 dhcpd: pool response: 10 leases
```


When a client requests an IP address, you will see messages like this in /var/log/messages:

```
Jul  4 11:59:26 da10 dhcpd: DHCPREQUEST for 192.168.5.16
from 00:04:ac:d6:58:96 via eth0: ignored (not
authoritative).
Jul  4 11:59:30 da10 dhcpd: DHCPREQUEST for 192.168.5.16
from 00:04:ac:d6:58:96 via eth0: ignored (not
authoritative).
Jul  4 11:59:36 da10 dhcpd: pool 800e4138 10.0.0/24 total
20  free 10  backup 10  lts 0
Jul  4 11:59:36 da10 dhcpd: DHCPDISCOVER from
00:04:ac:d6:58:96 via eth0
Jul  4 11:59:36 da10 dhcpd: DHCPREQUEST for 10.0.0.111
(10.0.0.12) from 00:04:ac:d6:58:96 via eth0: lease owned by
peer
Jul  4 11:59:37 da10 dhcpd: DHCPOFFER on 10.0.0.110 to
00:04:ac:d6:58:96 (linux-5ncs) via eth0
```

In this case, the client send a DHCPREQUEST in order to get same the IP address as the last time (192.168.5.16). This address is not available from a range definition, so the server refuses to offer this address. The the client send a DHCPDISCOVER to detect a DHCP server.

The next message from the client is a DHCPREQUEST for the IP address 10.0.0.111, this address is provided from the secondary server (“lease owned by peer”). The last message is the DHCPOFFER from the secondary server.

On the secondary server, the corresponding messages look like this:

```
Jul  4 12:00:20 da12 dhcpd: DHCPREQUEST for 149.44.85.16
from 00:04:ac:d6:58:96 via eth1: ignored (not
authoritative).
Jul  4 12:00:25 da12 dhcpd: DHCPREQUEST for 149.44.85.16
from 00:04:ac:d6:58:96 via eth1: ignored (not
authoritative).
Jul  4 12:00:30 da12 dhcpd: pool 800e40f8 10.0.0/24 total
20  free 10  backup 10  lts 0
Jul  4 12:00:30 da12 dhcpd: DHCPDISCOVER from
00:04:ac:d6:58:96 via eth1
Jul  4 12:00:31 da12 dhcpd: DHCPOFFER on 10.0.0.111 to
00:04:ac:d6:58:96 (linux-5ncs) via eth1
Jul  4 12:00:31 da12 dhcpd: DHCPREQUEST for 10.0.0.111
(10.0.0.12) from 00:04:ac:d6:58:96 (linux-5ncs) via eth1
Jul  4 12:00:31 da12 dhcpd: DHCPACK on 10.0.0.111 to
00:04:ac:d6:58:96 (linux-5ncs) via eth1
```

On this server, the DHCPOFFER message is printed as this server offers the IP address. The final message is DHCPACK from the client.

The DHCP log file /var/lib/dhcp/db/dhcpd.leases contains information like this:

```
failover peer "digitalairlines" state {
  my state recover at 2 2006/07/04 09:57:39;
  partner state unknown-state at 2 2006/07/04 09:57:39;
}

failover peer "digitalairlines" state {
  my state recover at 2 2006/07/04 09:57:39;
  partner state recover at 2 2006/07/04 09:57:39;
}
failover peer "digitalairlines" state {
  my state recover-done at 2 2006/07/04 09:57:56;
  partner state recover at 2 2006/07/04 09:57:39;
}

failover peer "digitalairlines" state {
  my state recover-done at 2 2006/07/04 09:57:56;
  partner state recover-done at 2 2006/07/04 09:57:39;
}

failover peer "digitalairlines" state {
  my state normal at 2 2006/07/04 09:57:56;
  partner state recover-done at 2 2006/07/04 09:57:39;
}
failover peer "digitalairlines" state {
  my state normal at 2 2006/07/04 09:57:56;
  partner state normal at 2 2006/07/04 09:57:39;
}
lease 10.0.0.120 {
  starts 2 2006/07/04 09:57:56;
  tstp 2 2006/07/04 09:57:56;
  binding state backup;
}
lease 10.0.0.119 {
  starts 2 2006/07/04 09:57:56;
  tstp 2 2006/07/04 09:57:56;
  binding state backup;
}
...
```

```
...
lease 10.0.0.120 {
  starts 2 2006/07/04 09:57:56;
  tstp 2 2006/07/04 09:57:56;
  tsfp 2 2006/07/04 09:57:56;
  binding state backup;
}
lease 10.0.0.119 {
  starts 2 2006/07/04 09:57:56;
  tstp 2 2006/07/04 09:57:56;
  tsfp 2 2006/07/04 09:57:56;
  binding state backup;
}

...
lease 10.0.0.111 {
  starts 2 2006/07/04 10:00:31;
  ends 2 2006/07/04 10:30:31;
  tstp 2 2006/07/04 09:57:56;
  tsfp 2 2006/07/04 10:45:31;
  cltt 2 2006/07/04 10:00:31;
  binding state active;
  next binding state expired;
  hardware ethernet 00:04:ac:d6:58:96;
  uid "\001\000\004\254\326X\226";
}
```


The file /var/lib/dhcp/db/dhcpd.leases on the secondary contains information like the following:

```
failover peer "digitalairlines" state {
  my state recover at 2 2006/07/04 09:58:50;
  partner state unknown-state at 2 2006/07/04 09:58:50;
  mclt 0;
}

failover peer "digitalairlines" state {
  my state recover at 2 2006/07/04 09:58:50;
  partner state recover at 2 2006/07/04 09:58:50;
  mclt 1800;
}
failover peer "digitalairlines" state {
  my state recover-done at 2 2006/07/04 09:58:51;
  partner state recover at 2 2006/07/04 09:58:50;
  mclt 1800;
}

failover peer "digitalairlines" state {
  my state recover-done at 2 2006/07/04 09:58:51;
  partner state recover-done at 2 2006/07/04 09:58:50;
  mclt 1800;
}
failover peer "digitalairlines" state {
  my state normal at 2 2006/07/04 09:58:51;
  partner state recover-done at 2 2006/07/04 09:58:50;
  mclt 1800;
}

failover peer "digitalairlines" state {
  my state normal at 2 2006/07/04 09:58:51;
  partner state normal at 2 2006/07/04 09:58:50;
  mclt 1800;
}
lease 10.0.0.120 {
  starts 2 2006/07/04 09:57:56;
  tsfp 2 2006/07/04 09:57:56;
  binding state backup;
}
```


When the secondary server fails (e.g. the server is shut down) and recovers later again, messages like the following will appear in /var/log/messages on the primary:

```
Jul  4 12:16:25 da10 dhcpd: peer digitalairlines:
disconnected
Jul  4 12:16:25 da10 dhcpd: failover peer digitalairlines:
I move from normal to communications-interrupted
Jul  4 12:20:06 da10 dhcpd: failover peer digitalairlines:
peer moves from normal to normal
Jul  4 12:20:06 da10 dhcpd: failover peer digitalairlines:
I move from communications-interrupted to normal
Jul  4 12:20:06 da10 dhcpd: pool 800e4138 10.0.0/24 total
20  free 10  backup 9  lts 0
```

On the secondary, the startup messages look like this:

```
Jul  4 12:21:01 da12 dhcpd: failover peer digitalairlines:
I move from normal to startup
Jul  4 12:21:01 da12 dhcpd: failover peer digitalairlines:
peer moves from normal to communications-interrupted
Jul  4 12:21:01 da12 dhcpd: failover peer digitalairlines:
I move from startup to normal
Jul  4 12:21:01 da12 dhcpd: failover peer digitalairlines:
peer moves from communications-interrupted to normal
Jul  4 12:21:01 da12 dhcpd: pool 800e40f8 10.0.0/24 total
20  free 10  backup 9  lts 0
```

In order to not get confused about the time stamps, make sure that you synchronize the time stamps of all your servers using the network time protocol (ntp).

The client log file /var/lib/dhcpcd/dhcpcd-eth0.info does only contain the information about the client's configuration and the DHCP server that provided the information:


```
IPADDR=10.0.0.111
NETMASK=255.255.255.0
NETWORK=10.0.0.0
BROADCAST=10.0.0.255
GATEWAY=10.0.0.254
DOMAIN='digitalairlines.com'
DNS=10.0.0.254
DHCPSID=10.0.0.12
DHCPGIADDR=0.0.0.0
DHCPSIADDR=0.0.0.0
DHCPCHADDR=00:04:AC:D6:58:96
DHCPSHADDR=00:04:AC:D6:55:F4
DHCPSNAME=''
LEASETIME=1800
RENEWALTIME=900
REBINDTIME=1575
INTERFACE='eth0'
CLASSID='Linux 2.6.16.20-0.12-default i686'
CLIENTID=00:04:AC:D6:58:96
```

Objective 7 Troubleshoot DHCP

The dhcp-tools package contains the two most important tools for troubleshooting DHCP. They will be explained in this objective:

- dhcping
- dhcpdump

dhcping

You can use the **dhcping** utility to check if the DHCP server responds.

You can specify

- The address the client wants to reserve (-c)
- The server to query (-s)
- The hardware address of the client (-h)

If the DHCP server responds, a message similar to the following appears:

```
da51:~ # dhcping -c 10.0.0.5 -s 10.0.0.51 -h 00:90:27:51:47:A9
Got answer from: 10.0.0.51
```

If the server does not respond, the message looks like the following:

```
da51:~ # dhcping -c 10.0.0.5 -s 10.0.0.200 -h 00:90:27:51:47:A9
no answer
```

Subsequently, the log file of the server will contain the following entries:

```
Nov 25 17:05:56 da51 dhcpd: DHCPREQUEST for 10.0.0.5 from
00:90:27:a4:96:1f via eth0
Nov 25 17:05:56 da51 dhcpd: DHCPACK on 10.0.0.5 to 00:90:27:a4:96:1f via
eth0
Nov 25 17:05:56 da51 dhcpd: DHCPRELEASE of 10.0.0.5 from 00:90:27:a4:96:1f
via eth0 (not found)
```

dhcpdump

You can use the **dhcpdump** utility to analyze the communication between the client and the server.

For this purpose, the output of tcpdump is piped to dhcpdump:

```
da51:~ # tcpdump -lenx -s 1500 port bootps or port bootpc | dhcpdump
```

The output shows the packets that were exchanged between the DHCP server and the DHCP client in a legible form, thus facilitating the analysis and troubleshooting.

Exercise 2-4 Troubleshoot DHCP

In this exercise, you troubleshoot DHCP.

You will find this exercise in the workbook.

(End of Exercise)

Summary

Objective	Summary
1. Understand How DHCP Works	DHCP communication includes the following steps: 1. On startup, the computer broadcasts a message to all the machines in the network to find a DHCP server. 2. A DHCP server answers the broadcasting machine’s request by offering an IP address and possible additional information. 3. The client chooses one of the responses it has received and sends a request to the corresponding DHCP server for the IP address. 4. The DHCP server, in turn, confirmós this information and the client is allowed to use the IP address. When the validity period of the IP address expires, the client sends a request to the server again and the server can either extend the duration of the IP address or prompt the client to request the assignment of a new address. If the client no longer needs the IP address, it sends a relevant notification to the DHCP server and the server can reassign the address.

Objective	Summary
1. Understand How DHCP Works (*continued*)	There are two ways IP addresses can be assigned with DHCP: ■ Static assignment. The administrator sets an IP address for a specific MAC address. ■ Dynamic assignment. The DHCP server assigns the host an IP address from an address range.

Objective	Summary
2. Configure the DHCP Server	To configure a Linux machine as a DHCP server, the following packages have to be installed: ■ **dhcp** ■ **dhcp-server** The configuration is made in the following files: ■ **/etc/sysconfig/dhcpd** ■ **/etc/dhcpd.conf** The file /etc/sysconfig/dhcpd contains an entry for the network interface on which the DHCP server listens. The file /etc/dhcpd.conf contains a number of entries that describe the network topology, influence the behavior of the server, or contain information transmitted to the clients. The entries in the file /etc/dhcpd.conf belong to two categories: ■ Parameter statements ■ Declarations Clients can be separated into classes and are treated differently depending on what class they are in. The records of the IP addresses already given are written to the file /var/lib/dhcp/db/dhcpd.leases.

Objective	Summary
3. Configure DHCP Clients	To configure a Linux host as a DHCP client software packages, like the dhcpd must be installed on the machine (included in the standard installation). The configuration file for each network interface is located in the directory **/etc/sysconfig/network/** This file contains the values with which a network card should be activated. Options for the DHCP client are stored in the file **/etc/sysconfig/network/dhcp**
4. Configure a DHCP Relay Server	Because routers do not route packets addressed to the broadcast address of the network, the DHCP server must be accessible for the DHCP clients in the local network. Instead of implementing a DHCP server in every local network, a DHCP relay server can be installed on the routers for the purpose of receiving the client broadcasts via one interface and forwarding it to the central server via the other interface. The respective package is dhcp-relay. The configuration of the DHCP relay is done in the file **/etc/sysconfig/dhcrelay**

Objective	Summary
5. Use DHCP and Dynamic DNS	If a client assigns variable IP addresses via DHCP, the DNS server must be informed about the host name and the IP address. This can be done in two ways: ■ The client transmits its name to the DHCP server and gets an IP address from the DHCP server. The DHCP server transmits the host name and the IP address to the DNS server. ■ The client gets its name and IP address from the DHCP server. The DHCP server transmits the host name and IP address to the DNS server. To enable interaction between the DHCP server and DNS server, you have to modify the following files ■ **/etc/named.conf** ■ **/etc/dhcpd.conf** Since the DNS server should only accept changes from the authorized DHCP server, the updates must be protected cryptographically. The program **dnssec-keygen** generates the key.

Objective	Summary
5. Use DHCP and Dynamic DNS (*continued*)	The name server first saves dynamic entries in files with the extension **.jnl**. At regular intervals, the changes are transferred from the .jnl file to the zone file by overwriting the old zone file. New entries should not be added to the zone file with an editor; use the **nsupdate** utility.
6. Configure DHCP Failover	The failover protocol allows two DHCP servers to share a common address pool. If one server fails, the other server will continue to renew leases out of the pool and will allocate new addresses. The failover protocol defines a primary server role and a secondary server role. In order to configure failover, you need to write a peer declaration in the file /etc/dhcpd.conf that configures the failover protocol, and you need to write peer references in each pool declaration for which you want to do failover.

Objective	Summary
7. Troubleshoot DHCP	The **dhcp-tools** package contains a number of troubleshooting tools. You can use the **dhcping** utility to check if the DHCP server responds. You can use the **dhcpdump** utility to analyze the communication between the client and the server. For this purpose, the output of tcpdump is piped to dhcpdump.

SECTION 3 Manage OpenLDAP

OpenLDAP is the most popular open source LDAP suite. It provides not only the LDAP server itself, but also applications and tools to control and query the server and to develop LDAP-based software.

Objectives

1. Understand the Basics of LDAP
2. Install and Set Up an OpenLDAP Server
3. Add Entries to the LDAP Server
4. Query Information from the LDAP Server
5. Modify and Delete Entries of the LDAP Server
6. Activate LDAP Authentication
7. Replicate OpenLDAP Servers

Objective 1 Understand the Basics of LDAP

A directory is a specialized database that is optimized for reading, browsing, and searching. Directories contain descriptive, attribute-based information and support sophisticated filtering.

Directories are tuned to give quick response to high-volume lookup or search operations. They can replicate information widely in order to increase availability and reliability, while reducing response time.

There are many different ways to provide a directory service. Different methods allow different kinds of information to be stored in the directory; place different requirements on how that information can be referenced, queried, and updated; and determine how it is protected from unauthorized access.

Some directory services are local, providing service to a restricted context (such as the finger service on a single machine). Other services are global, providing service to a much broader context (such as the entire Internet).

Directory services can be used for many different purposes. Very often they are used as databases for user authentication. By default, SUSE Linux Enterprise Server 10 uses OpenLDAP for user management and some configuration purposes.

LDAP stands for *Lightweight Directory Access Protocol*. As the name suggests, it is a lightweight protocol for accessing directory services called “DAP”.

DAP was developed in 1988 by the CCITT (*International Telegraph and Telephone Consultative Committee*). It should manage access to information stored in a X.500 directory service. There are two more Protocols involved:

- DSP. (Directory Service Protocol) Manages the communication between servers.
- DISP. (Directory Information Shadowing Protocol) Transports the results of a database request.

X.500 is based on a complete ISO/OSI stack. Clients that use IP are only able to access a server that needs a complete ISO/OSI stack via a gateway. Because of this, DAP could not win recognition.

In 1995 Yeong, Howes, and Kill, from the University of Michigan, developed LDAP. Only the DAP functionalities of X.500 are reused and some new features were added.

LDAP runs over TCP/IP or other connection-oriented transfer services. Because of this it can be integrated easily into an existing network.

LDAP stores information in *objects* that can be associated to *object classes*. The classes determine which attributes an object can/must have. By including so-called *schemas*, you are able to access pre-defined object classes.

OpenLDAP ships with some basic schema files located in the directory /etc/openldap/schema/.

Frequently used object classes are, for example:

- **alias**
- **country**
- **locality**
- **organization**

- **organizationalUnit**
- **organizationalRole**
- **person**

Each object is a collection of *attributes* that has a globally unique distinguished name (DN). The DN is used to refer to the entry. Each of the entry's attributes has a type and one or more values.

The attributes are typically mnemonic strings, like “cn” for common name, or “mail” for email addresses. The syntax of values depends on the attribute type.

For example, a cn attribute might contain the value “Geeko Chameleon.” A mail attribute might contain the value “geeko@digitalairlines.com.” A **jpegPhoto** attribute might contain a photograph in the JPEG (binary) format.

Some attributes of these classes include:

- **dn** (distinguished name). Unique name for the object.
- **objectClass**. Class the object belongs to.
- **cn** (common name). Name (e.g., name of a user)
- **sn** (surname).
- **o** (organization name). Name of an organisation (e.g., company name)
- **ou** (organizational unit). Name of an organisational unit (e.g., name of a department)
- **l** (locality name). Location
- **streetadress**. Street
- **c** (country).
- **postalcode**
- **telephonenumber**

- **mail**. Email address
- **description**. Description of the object

Object classes that LDAP uses for user authetication include

- **posixAccount**
- **shadowAccount**
- **posixGroup**

Some of the attributes that are used in these classes include:

- **uid**. Login of the user
- **uidNumber**. Numerical user ID
- **gid**. Group name
- **gidNumber**. Numerical group ID
- **homeDirectory**. Path of the home directory
- **loginShell**. Path of the login shell
- **shadowLastChange**. Date of the last changing of the password

In LDAP, objects can be arranged in a hierarchical tree structure. You can distinct between two kinds of objects:

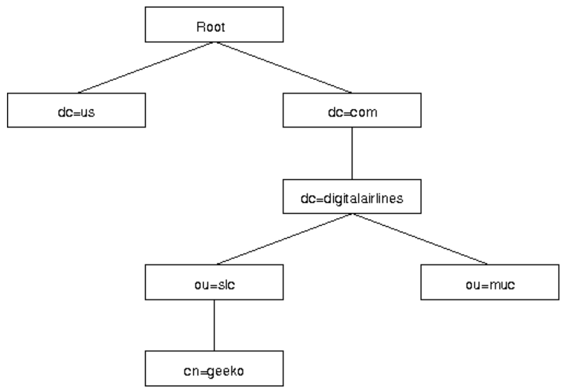
- **Container objects**. Container objects are objects, that can have subordinate objects. There are five classes of container objects:
    - Root
    - c (country)
    - o (organization)
    - ou (organizational unit)
    - dc (domain component)
- **Leaf objects**. Leaf objects cannot have any subordinate objects. They are the end of a tree branch. For example:
    - cn (common name)

- uid (user ID)

If you use LDAP for user management, the structure (DIT, *Directory Information Tree*) normally reflects:

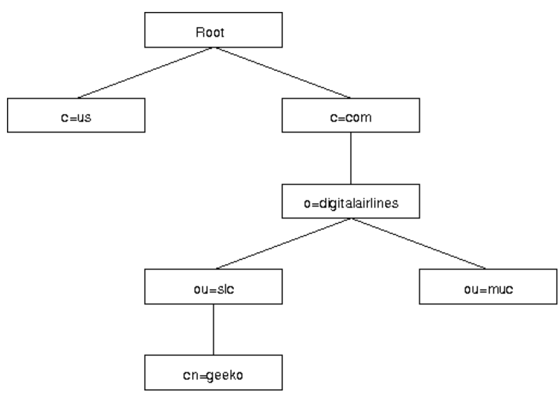
- the organizational structure of the company or organization

Figure 3-1



- or can be similar to the domain system

Figure 3-2



Under the root of the tree are the country, organization, organizational unit and leaf objects (such as users).

An object of the tree is referenced by its DN, which is constructed by taking the name of the entry itself (called the relative distinguished name or RDN) and concatenating the names of its ancestor objects.

For example, the entry for Geeko Chameleon in the example above has a DN of **cn=geeko,ou=slc,dc=digitalairlines,dc=com**.

Objective 2 Install and Set Up an OpenLDAP Server

To install and set up an OpenLDAP server, you need to do the following:

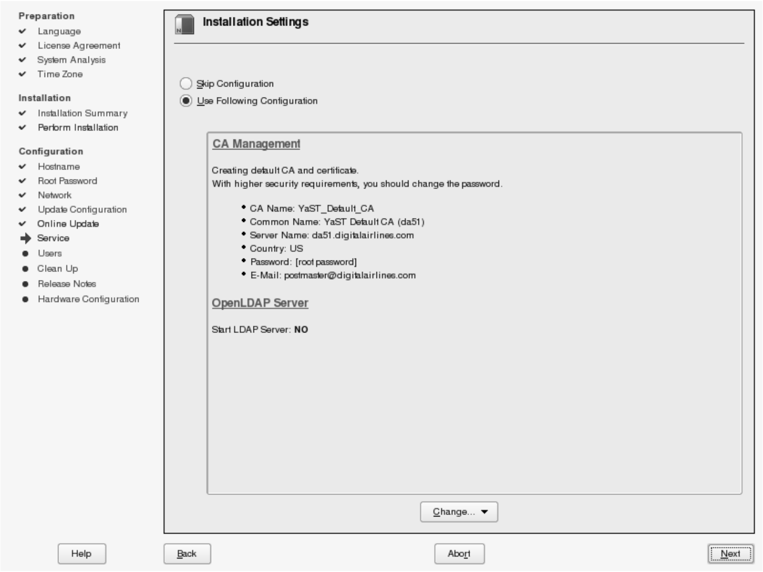
- Install the Required Software and Start the Server
- Configure OpenLDAP with YaST
- Edit the OpenLDAP Configuration Files

Install the Required Software and Start the Server

Normally, you can set up an OpenLDAP server with YaST during the installation process of SUSE Linux Enterprise Server 10.

Select **OpenLDAP Server** in the Installation Settings dialog.

Figure 3-3

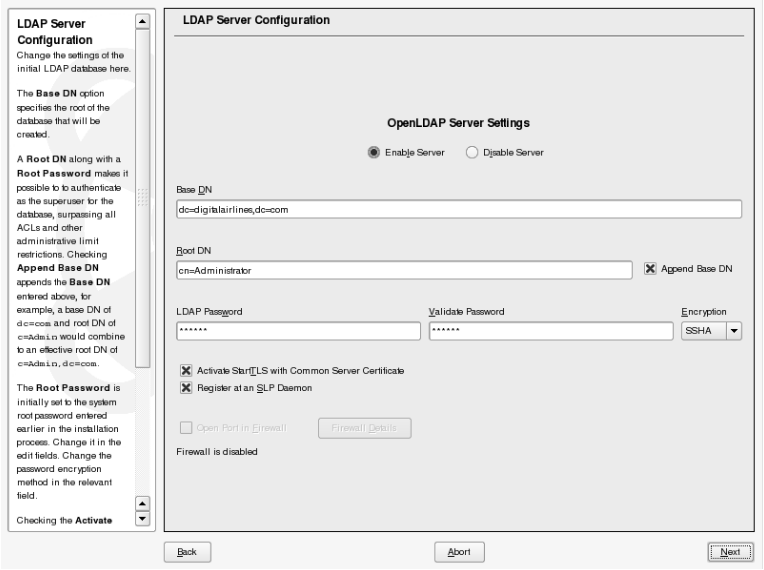


The important values are entered automatically in the text fields. A warning informs you that any change will disable the automatic generation of base DN, root DN, and the LDAP password.

Figure 3-4

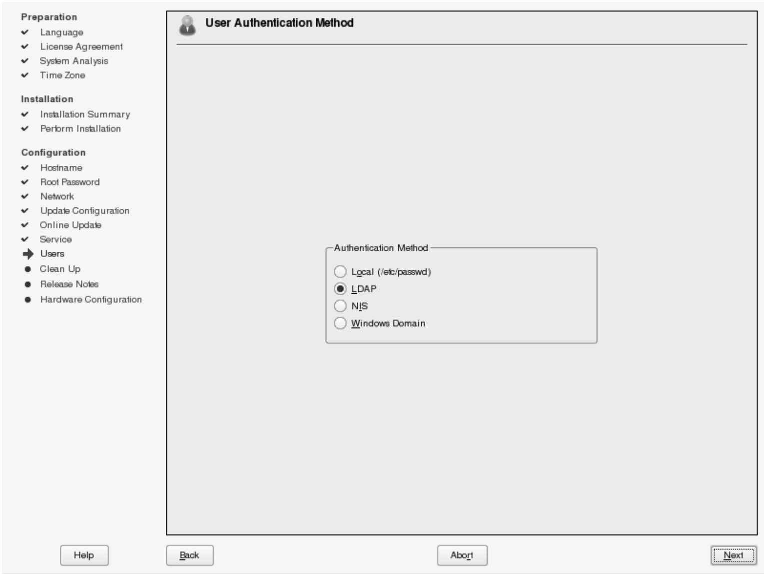
Select **OK** and enable the OpenLDAP server by selecting **Enable Server**.

Figure 3-5



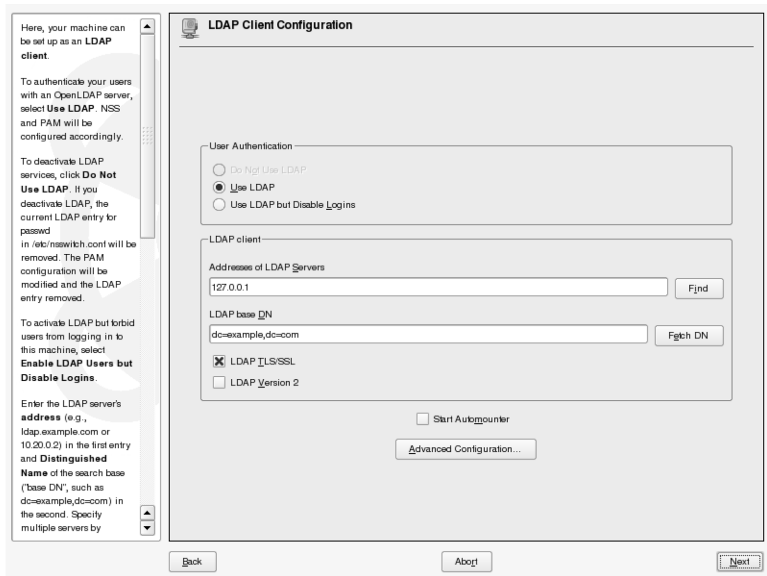
If you want to authenticate via LDAP, select **LDAP** in the **User Authentication Method** dialog.

Figure 3-6



In the next dialog, you can enter the configuration of your LDAP client.

Figure 3-7



Here you can enter the following information:

- **User Authentication**. Select the kind of user authentication:
  - **Do Not Use LDAP**. LDAP is not used for user authentication.
  - **Use LDAP**. LDAP is used for user authentication.
  - **Use LDAP but Disable Logins**. LDAP is used for user authentication, but the users are not able to log in to this machine anymore.
- **Addresses of LDAP Servers**. Enter the name or the IP address of your LDAP server here. If the IP of your LDAP server is provided via SLP, you can select **Find**.

- **LDAP base DN**. Enter the DN of the root of your LDAP tree here. Select **Fetch DN** if you want to browse the LDAP tree of your LDAP server.
- **LDAP TLS/SSL**. If your LDAP server supports encryption, activate this option.
- **LDAP Version 2**. The current LDAP protocol version is 3. If you want to use the older version 2, select this option.
- **Start Automounter**. Starts the automounter daemon which mounts directories automatically when they are used.

Configure OpenLDAP with YaST

However, if you chose not to install the server during installation, you can set up an LDAP server by installing the following software packages with YaST:

- openldap2
- openldap2-client
- pam_ldap
- nss_ldap

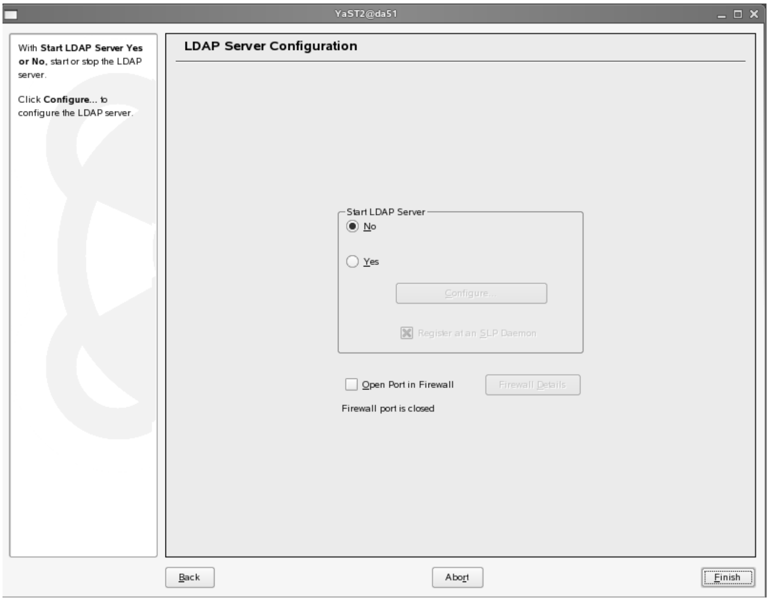
In YaST two modules concerning OpenLDAP are available:

- YaST Module for the LDAP Server
- YaST Module for the LDAP Client

YaST Module for the LDAP Server

If you select **LDAP Server** in the YaST **Network Services** section, but your LDAP server is currently not running, the following dialog appears.

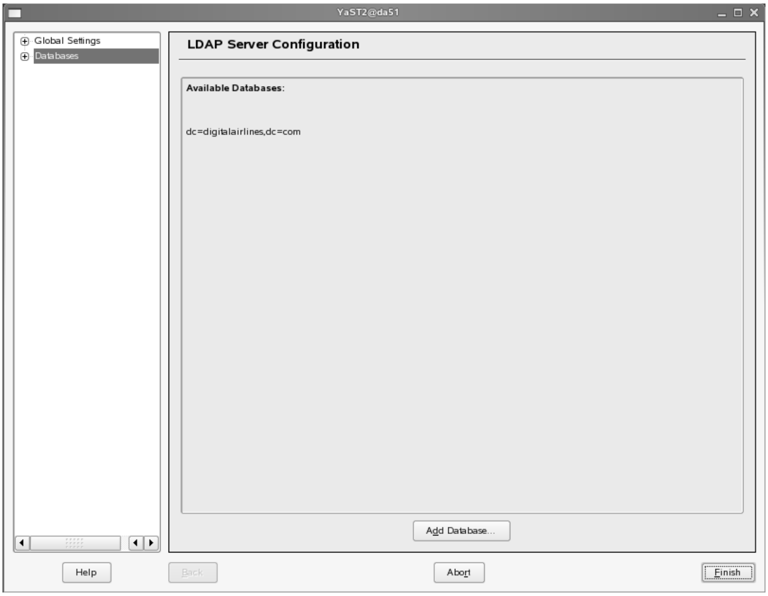
Figure 3-8



Select **Yes** to start and configure the LDAP server. You can also activate the option to register the LDAP server at a SLP daemon or to open the LDAP port in the firewall.

Select **Configure** to configure the LDAP server.

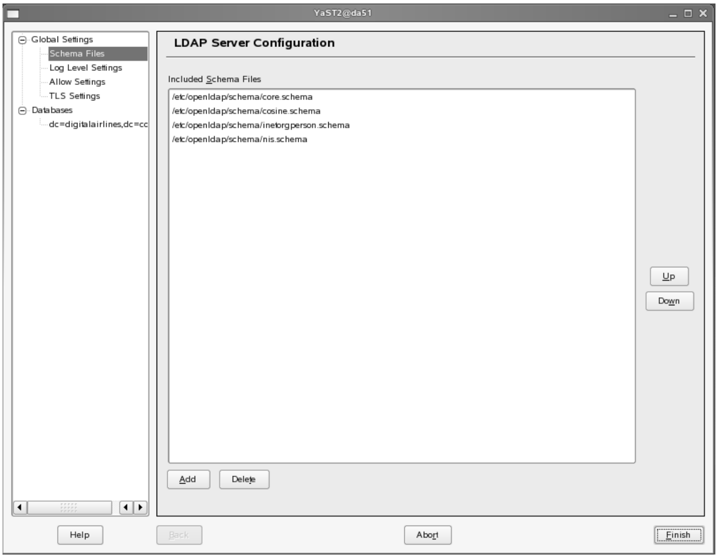
Figure 3-9



Open the **Global Settings** submenu in the left frame. The following settings can be configured:

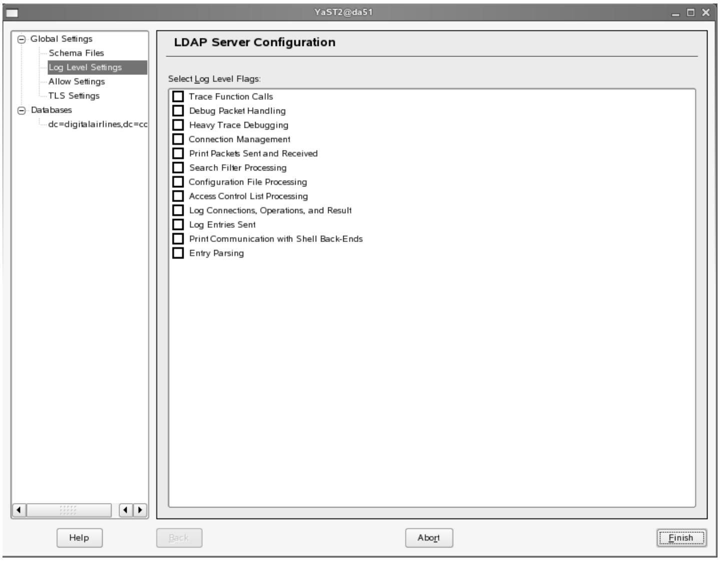
- **Schema Files**. Add or remove LDAP schema files using **Add** or **Delete**.

Figure 3-10



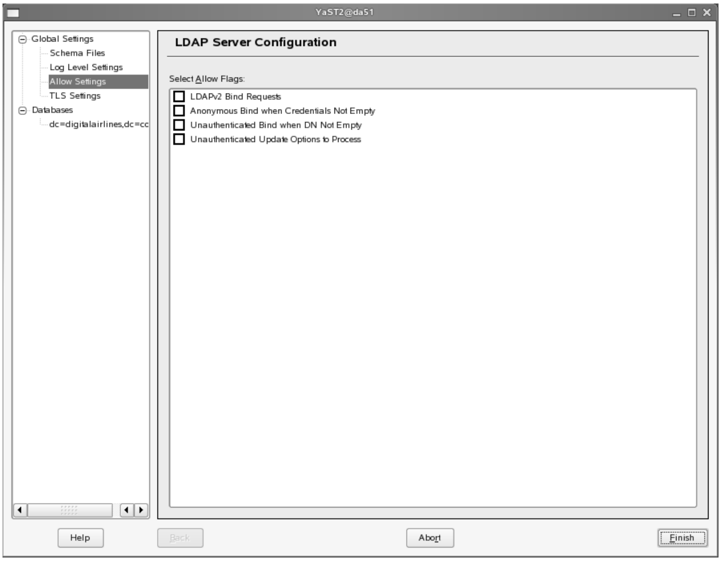
- **Log Level Settings**. Select the information you want to log.

Figure 3-11



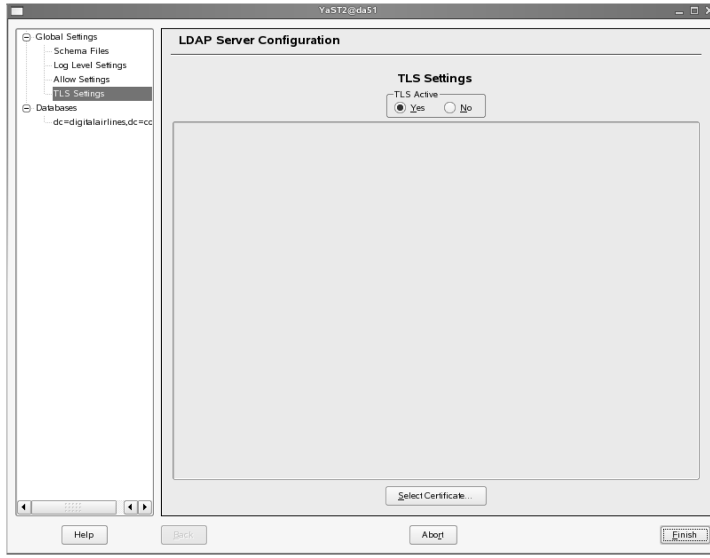
- **Allow Setttings**. Select options concerning access to stored information.

Figure 3-12



- **TLS Settings**. Activate or Deactivate TLS.

Figure 3-13



If you select **Select Certificate**, you can import a certificate.

To create a new certificate use the YaST module Security and **Users > CA Management**.

If you select **Databases** in the left frame, you can add a new database by selecting Add Database. The following dialog appears.

Figure 3-14

In this dialog you have to enter the following information:

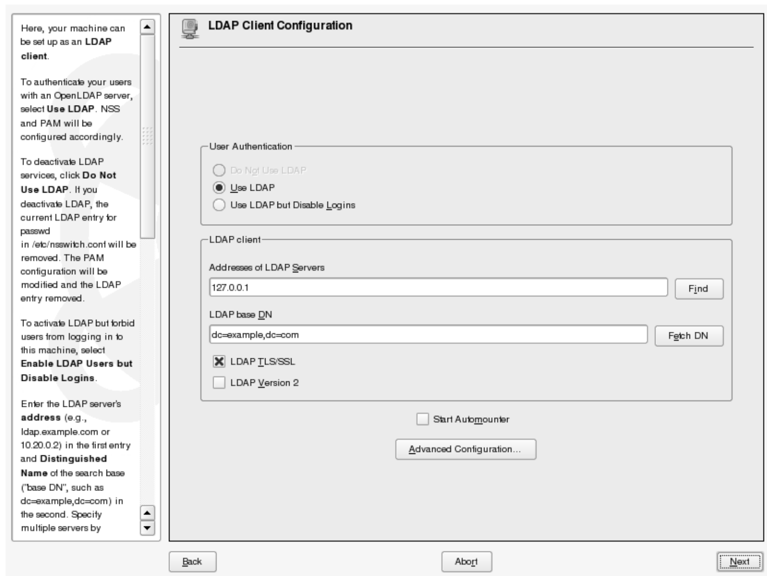
- **Base DN**. DN of the root of the LDAP tree.
- **Root DN**. DN of the administrator. If you activate **Append Base DN**, the content of **Base DN** is appended to the input of **Root DN** automatically.
- **LDAP Password**, **Validate Password**. The administrator's password. From the **Encyption** menu you can select the encryption method that should be used to encrypt the password.
- **Database Directory**. Path to the directory the database is stored in.

Most of the information of the YaST LDAP server are stored in the file /etc/sysconfig/openldap.

YaST Module for the LDAP Client

When selecting **LDAP Client** in the YaST **Network Services** section, the following dialog appears as described above.

Figure 3-15



This information is stored in the file /etc/sysconfig/ldap. Three variables are used:

BASE_CONFIG_DN. DN of the root of the LDAP tree.

BIND_DN. The DN of the administrator.

FILE_SERVER.

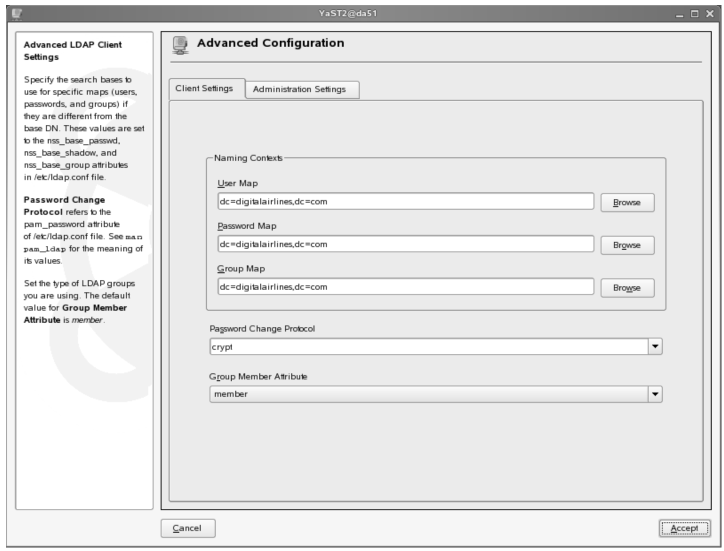
Selecting the **Advanced Configuration** button a dialog with two tabs appears:

- Client Settings
- Administrator Settings

Client Settings

In this dialog you can specify the search base for users, passwords, and groups.

Figure 3-16



The items of the dialog are:

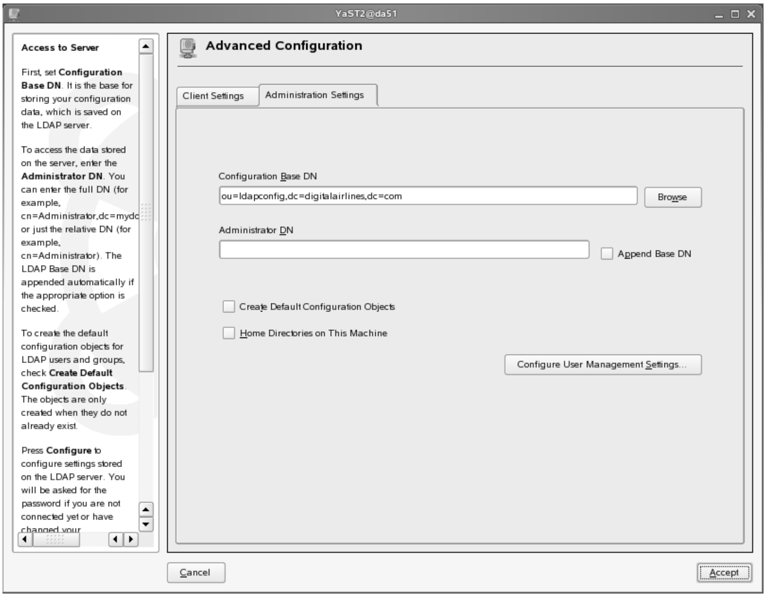
- **User Map**. Enter the DN where users are stored.
- **Password Map**. Enter the DN where passwords are stored.
- **Group Map**. Enter the DN where groups are stored.
- **Password Change Protocol**. The password change protocol. See **man 5 pam_ldap** to get more information about the possible values.
- **Group Member Attribute**. Set the type of the LDAP groups you are using. The value can depend on the schema files you are using. The default is **member**.

This information is stored in the file /etc/ldap.conf.

Administrator Settings

In this dialog, you specify some administrator settings.

Figure 3-17

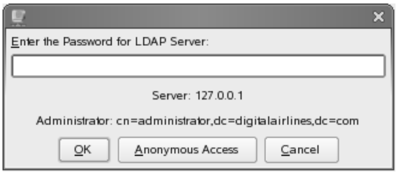


Enter the following information:

- **Configuration Base DN**. The configuration data are also stored in the LDAP directory. Here you can enter the base DN for all configuration information.
- **Administrator DN**. The DN of the LDAP administrator.
- **Create Default Configuration Objects**. To create the default configuration objects for users and groups, select this option.
- **Home Directories on This Machine**. If home directories of users should be stored on your machine, select this option. This option is inly information for the YaST user module that manages user home directories.

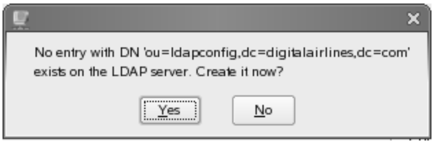
If you want to configure the user management settings stored on the LDAP server (e.g., the next free UID), select **Configure User Management Settings**. You are asked to enter the administrator's LDAP password.

Figure 3-18



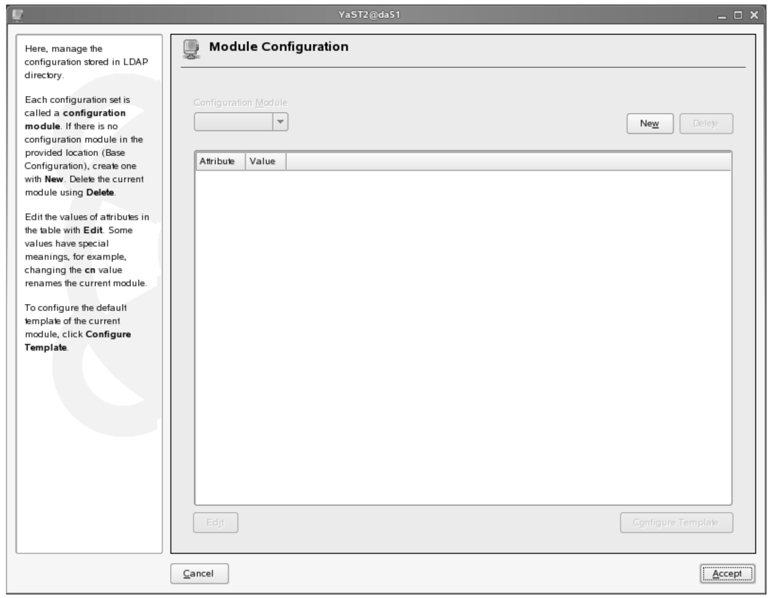
After you enter the administrator password, YaST determines if the configuration base DN exists in the LDAP directory. If not, it asks if the entry should be created.

Figure 3-19



Select **Yes** and the Module Configuration dialog appears.

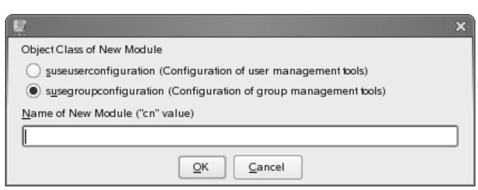
Figure 3-20



A configuration module is a set of configurations. You can manage configuration modules for user and group configurations.

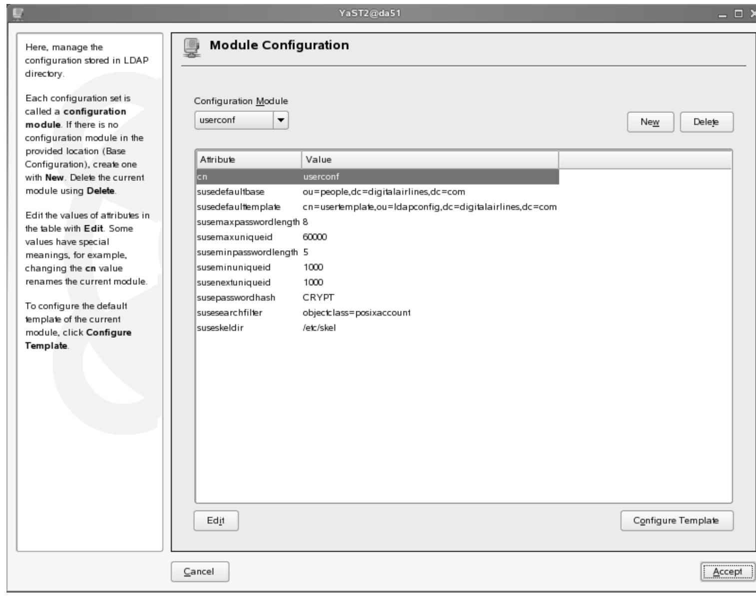
Select **New** to create a new module.

Figure 3-21



Select whether you want to create a module for user configuration or for group configuration. You also have to enter the common name (cn) of the module.

Figure 3-22



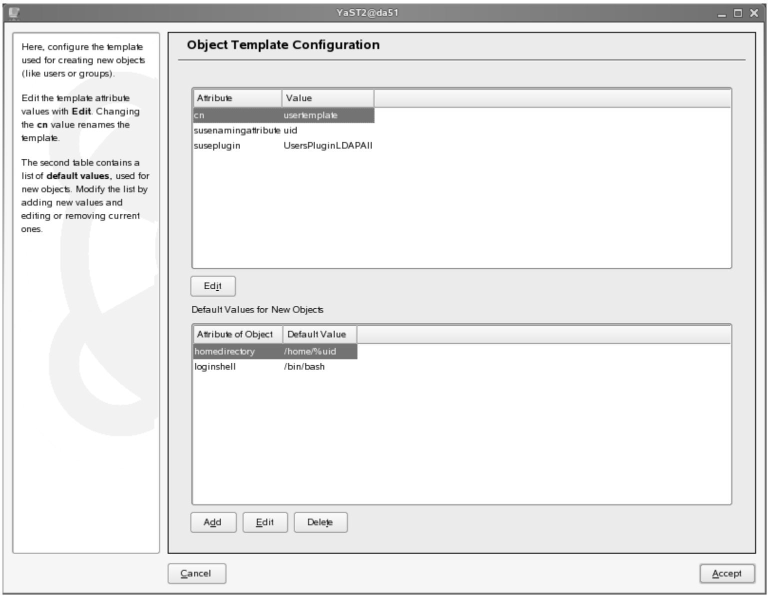
A template for the user or group module is shown. You can find the listed attributes in the YaST schema file (/etc/openldap/schema/yast.schema).

To edit a value of an attribute, select a line from the list and then select **Edit**.

Figure 3-23

In the line of the attribute **susedefaulttemplate**, a template for new users or groups is specified. If you want to edit this template, select **Configure Template**.

Figure 3-24



In the upper frame you see a list of attributes and their values. Edit the template attribute values with **Edit**.

In the lower frame you can specify default values for attributes. Use the three buttons below as follows:

- **Add**. Add a new default value.
- **Edit**. Edit the selected default value.
- **Delete**. Delete the selected default value.

Exercise 3-1 Set Up OpenLDAP with YaST

In this exercise, you set up an OpenLDAP server and client using YaST.

You will find this exercise in the workbook.

(End of Exercise)

Edit the OpenLDAP Configuration Files

The configuration files for OpenLDAP are located in the directory /etc/openldap/. The directory contains two configuration files:

- Configure the OpenLDAP Server with slapd.conf
- Configure the LDAP Clients with ldap.conf

Configure the OpenLDAP Server with slapd.conf

The default version of the configuration file /etc/openldap/slapd.conf has only a few options. Most of them do not need to be changed:

- **include**. The first entry includes a schema file. In this schema file some object classes and their attributes are defined. You can use these object classes to build a database.

  For simple example configurations, only the object classes of the *core* schema needs to be included.
- **referral**. Specifies the referral to pass back when the slapd cannot find a local database to handle a request.
- **pidfile**. Path to the file that stores the process ID.
- **argsfile**. Path to the file that stores the server's command line options.
- **modulepath**. Backend modules that are loaded dynamically. With the **modulepath** you specify a list of directories to search for loadable modules.
- **moduleload**. Specifies the filename or the absolute path of a dynamically loadable module.
- **security**. Specifies a set of security strength factors to require. The directive may be specified globally and/or per-database. Possible security strength factors are:
  - **ssf**. Specifies the overall security strength factor.

- **transport**. Specifies the transport security strength factor.
- **tls**. Specifies the TLS security strength factor.
- **sasl**. Specifies the SASL security strength factor.
- **update_***. Specifies the security strength factors required for directory updates.
- **simple_bind**. Specifies the security strength factor required for simple username/password authentication.

For a more detailed description, read the section about sasl-secprops in the slapd.conf manpage (**man 5 slapd.conf**).

- **access to**. Grants access to a set of entries and/or attributes by one or more requesters.

  Syntax: **access to *objects* [by *requesters level control*]**

  The ***objects*** and the ***requesters*** are specified by their DNs.

  As ***requesters***, you also can use the asterisk for *everybody*. **self** means the owner object of the attribute. You can also specify a DN by **dn=*DN*** or a group by **group=*DN***.

  The ***level*** can be:

  - **read** (Read access)
  - **write** (Write access)
  - **auth** (Authentication needed)
  - **none** (No access)

  Example 1:

```
access to attrs=userpassword
  by self write
  by anonymous auth
  by * none

access to *
  by * read
```

Example 2:

```
access to *
  by dn="cn=Administrator,dc=digitalairlines,dc=com" write
  by * read
```

- **database**. Specifies the kind of back end. The following databases are supported:
  - **bdb**. Berkeley DB
  - **dnssrv**. DNS SRV
  - **hdb**. A hierarchical variant of bdb
  - **ldap**. Lightweight Directory Access Protocol (Proxy)
  - **ldbm**. Lightweight DBM
  - **meta**. Meta directory
  - **monitor**. Monitor backend
  - **passwd**. Read-only access to passwd
  - **perl**. Perl programmable backend
  - **shell**. External shell program
  - **sql**. SQL programmable backend
- **suffix "*DN*"**

  Specify the DN suffix of queries that will be passed to this backend database.

  Example:

```
suffix          "dc=digitalairlines,dc=com"
```

- **checkpoint *ops minutes***. Checkpoints are only tested after successful write operations. If ***ops*** operations have been completed or more than the specified number of ***minutes*** have passed, a new checkpoint is performed.

- **cachesize**. Size of in-memory cache of the LDBM backend database.
- **rootdn "*DN*"**

  This line sets the administrator of the LDAP server. You can also configure the domain components in this line.

  Example:

```
rootdn          "ou=slc,dc=digitalairlines,dc=com"
```

- **rootpw *password***

  This line specifies the password for the administrator. The default password secret must be changed.

  For security reasons, the password should be stored in an encrypted form. To create an encrypted password, use the following command:

  slappasswd -s *your_password*

  The command outputs a string that has to be copied into the configuration file.

```
da51:~ # slappasswd -s novell
{SSHA}5JHiJ3Q17MpxaNT95DqVg2u1VfjZyzHh
```

  The entry for the command rootpw looks like the following:

```
rootpw          "{SSHA}5JHiJ3Q17MpxaNT95DqVg2u1VfjZyzHh"
```

- **directory**. Path to the BDB or HDB database. This directory must exist before the server is started.
- **index**. Specifies the indices to maintain. This increases the speed of the search.

  Syntax: **index *attributes types***

Possible ***types*** are:

- **pres**. Allows filters to be used that ask if the attribute is present in an entry (cn=*).
- **eq**. Allows filters to be used that ask if an attribute has an exact value. It includes presence, so it is not necessary to index something as pres,eq.
- **approx**. Allows filters to be used that ask if an attribute value is "similar to" something.
- **sub**. Allows filters to be used that do substring searches on an attribute's values.
- **none**.

After finishing the configuration, you can start the server with the following command:

rcldap start

If you want to start the LDAP server automatically when the server boots, use the following command:

insserv ldap

Configure the LDAP Clients with ldap.conf

After you have changed the server configuration file, you need to change the client configuration. Two files are important:

- /etc/openldap/ldap.conf
- /etc/ldap.conf

/etc/openldap/ldap.conf

The file /etc/openldap/ldap.conf is the configuration file of the LDAP client.

The most important options are specified below:

- **host 127.0.0.1**

  This line sets the default server that LDAP clients should connect to.

- **base dc=digitalairlines,dc=com**

  This is the default directory search base that should be used by LDAP clients.

After a standard installation, there is one more line in /etc/ldap.conf:

```
TLS_REQCERT     allow
```

TLS_REQCERT specifies what checks, if any, to perform on server certificates in a TLS session. The level can be specified as one of the following keywords:

- **never**. The client will not request or check any server certificate.
- **allow**. The server certificate is requested. If no certificate or a bad certificate is provided, the session proceeds normally.
- **try**. The server certificate is requested. If no certificate is provided, the session proceeds normally. If a bad certificate is provided, the session is immediately terminated.
- **demand** or **hard**. The server certificate is requested. If no certificate or a bad certificate is provided, the session is immediately terminated. (default)

For more options, read the manual **man 5 ldap.conf**.

/etc/ldap.conf

The file /etc/ldap.conf is the configuration file for the LDAP nameservice switch library, the LDAP PAM module, and the shadow package. The file belongs to the RPM package pwdutils.

```
da51:~ # rpm -qf /etc/ldap.conf
pwdutils-3.0.7.1-17.10
da51:~ #
```

The most important options of /etc/ldap.conf are:

- **host**. Specifies the name(s) or IP address(es) of the LDAP server(s) to connect to.
- **base**. Specifies the default base distinguished name (DN) to use for searches.
- **ldap_version**. Specifies the version of the LDAP protocol; "*3*" stands for LDAPv3.
- **bind_policy**. Specifies what happens if the LDAP server is not reachable. The possible values are:
  - **hard**. nss_ldap will retry connecting to the software with exponential backoff. (default)
  - **soft**. Forbids nss_ldap from retrying failed LDAP queries.
- **binddn**. Specifies he distinguished name with which to bind to the directory server(s).

  The default is to bind anonymously.
- **bindpw**. Specifies the cleartext credentials ith which to bind. This option is only applicable when used with binddn above.
- **scope**. Specifies the search scope. Possible values are:
  - **sub**. subtree
  - **one**. one level
  - **base**. base object

The default scope is **sub**; **base** scope is almost never useful for nameservice lookups.

- **pam_password**. Specifies the password change protocol to use. The following protocols are supported:
  - **clear**. Change password using an LDAPModify request, replacing the userPassword value with the new cleartext password.
  - **clear_remove_old**, **nds** or **racf**. Change password using an LDAPModify request, first removing the userPassword value containing the old cleartext password, and then adding the userPassword value with the new cleartext password. This protocol is necessary for use with Novell NDS and IBM RACF.
  - **crypt**. Change password using an LDAPModify request, first generating a one-way hash of the new password using crypt(3) and then replacing userPassword value with the new hashed password.
  - **md5**. Change password using an LDAPModify request, first generating a one way hash of the new password using MD5 and then replacing userPassword value with the new hashed password.
  - **ad**. Change password using an LDAPModify request, using the Active Directory Services Interface (ADSI) password change protocol.
  - **exop**. Change password using the RFC 3062 password modify extended operation (only the new password is sent).
  - **exop_send_old**. Change password using the RFC 3062 password modify extended operation (both the old and new passwords are sent). This is the preferred choice when using the PADL XAD identity server.
- **ssl**. Specifies whether to use SSL/TLS or not. The following values are possible:
  - **on**

- **off** (default)
- **start_tls**

If **start_tls** is specified, then StartTLS is used rather than raw LDAP over SSL.

Not all LDAP client libraries support both SSL and StartTLS, and all related configuration options.

- **nss_map_attribute** ***from_attribute to_attribute***. This option may be specified multiple times, and directs nss_ldap to use the attribute ***to_attribute*** instead of the RFC 2307 attribute ***from_attribute*** in all lookups.

  For example: **nss_map_attribute uniqueMember member**

- **pam_login_attribute**. Specifies the attribute to use when constructing the attribute value assertion for retrieving a directory entry for a user's login name.

  The default is "uid", for compatibility with RFC 2307.

- **pam_filter**. Specifies a filter to use when retrieving user information. The user entry must match the **pam_login_attribute** attribute value as well as any filter specified here.

  For example: **pam_filter objectclass=posixAccount**

- **nss_base_*map***. Specify the search base, scope, and filter to be used for specific maps.

Note that map forms part of the configuration file keyword and is one of the following options **passwd**, **shadow**, **group**, **hosts**, **services**, **networks**, **protocols**, **rpc**, **ethers**, **netmasks**, **bootparams**, **aliases**, and **netgroup**.

The syntax of **basedn** and **scope** are the same as for the configuration file options of the same name, with the addition of being able to omit the trailing suffix of the base DN (in which case the global base DN will be appended instead).

For more options, read the manual **man 5 pam_ldap**.

Objective 3 Add Entries to the LDAP Server

OpenLDAP provides the command **ldapadd** to insert data that is in LDIF format into the directory. You can use files in the LDIF format to avoid specifying all values on the command line.

LDIF files contain the information that should be included in the directory service in a plain text format.

You can create a different file for each user you would like to add, but you can also include multiple user records in one file. An LDIF file has the following structure:

dn: ***DN***
objectclass: ***class***
attribute**:** ***value***
attribute**:** ***value***

For each object, you have to specify its distinguished name first. This must be unique and represent the position in the directory service tree.

It follows the object class(es) the object belongs to and one or more attributes. For example:

```
dn: dc=digitalairlines,dc=com
objectclass: domain
objectclass: top
o: digitalairlines

dn: ou=people,dc=digitalairlines,dc=com
objectclass: organizationalUnit
ou: people
```

The file creates two objects in the directory tree:

- The root object with the DN dc=digitalairlines,dc=com. Before you can add the first object, you have to create the root of the directory service tree.

- An object below the base entry for an organizational unit. The dn for this object is ou=people,dc=digitalairlines,dc=com.

Make sure that there are no empty spaces or tabs at the beginning or end of a line.

Because LDAP uses Unicode (UTF-8), special characters in LDIF files have to be coded into UTF-8, or they might not be evaluated. This means you need to edit the LDIF file with a Unicode editor, or convert the file later.

You can convert the file by entering the following command:

recode lat1..utf8 ***ldif_file***

To know which object classes are available and which attributes are mandatory and which are optional, you have to look into the schema file.

For example, in the schema file /etc/openldap/schema/core.schema, there is an object class *organization*:

```
objectclass ( 2.5.6.4 NAME 'organization'
        DESC 'RFC2256: an organization'
        SUP top STRUCTURAL
        MUST o
        MAY ( userPassword $ searchGuide $ seeAlso $
businessCategory $
                x121Address $ registeredAddress $
destinationIndicator $
                preferredDeliveryMethod $ telexNumber $
teletexTerminalIdentifier $
                telephoneNumber $ internationaliSDNNumber $
                facsimileTelephoneNumber $ street $
postOfficeBox $ postalCode $
                postalAddress $ physicalDeliveryOfficeName
$ st $ l $ description )
 )
```

Objects of this class must have a value for the *o* attribute.

The command to insert a data set that exists as an LDIF file looks like the following:

ldapadd -x -D *dn_of_the_administrator* -W -f *file*.ldif

You need to use the **-x** option because you haven't configured SASL authentication yet.

Use the option **-D** to specify who can access the directory. This should be rootdn, specified in the server configuration file.

Use the option **-W** to display a password prompt. Otherwise, you must enter the password directly at the command line, where it will be visible as plain text.

Finally, specify the LDIF file with the option **-f**.

If the LDIF file is called example.ldif, ldapadd should be run as follows:

```
da51:~ # ldapadd -x -D "cn=Administrator,dc=digitalairlines,dc=com" -W -f
example.ldif
Enter LDAP Password:
adding new entry "dc=digitalairlines,dc=com"

adding new entry "cn=people,dc=digitalairlines,dc=com"
```

The password that is asked for must fit the password specified at **rootpw** in /etc/openldap/slapd.conf.

The password is transfered in clear text via the network. It is recommend that you configure the LDAP server locally or to use ssh.

After you have set up the basic tree structure, you can add a user to the directory with an LDIF file similar to the following:

```
dn: uid=geeko,ou=people,dc=digitalairlines,dc=com
objectClass: posixAccount
objectClass: shadowAccount
objectClass: inetOrgPerson
uid: geeko
uidNumber: 1000
gidNumber: 100
cn: Geeko Chameleon
givenName: Geeko
sn: Chameleon
homeDirectory: /home/geeko
loginShell: /bin/bash
shadowMax: 99999
shadowWarning: 7
shadowInactive: -1
shadowMin: 0
shadowLastChange: 12609
```

This example LDIF file creates a user based on the default LDAP setup of SUSE Linux Enterprise Server 10. The attributes are listed below, along with an explanation of each:

- **uid: geeko.** This attribute sets the login name of the user.
- **uidNumber: 1000.** This attribute sets the numerical ID of the user.
- **gidNumber: 100.** This attribute sets the default group ID of the user. The value 100 belongs to the group users in a SUSE Linux Enterprise Server 10 installation.
- **cn: Geeko Chameleon.** This attribute sets the full name of the user.
- **givenName: Geeko.** This attribute sets the given name of the user.
- **sn: Chameleon.** This attribute sets the surname of the user.
- **homeDirectory: /home/geeko.** This attribute sets the path to the home directory of the user.

- **loginShell: /bin/bash.** This attribute sets the login shell of the user. The default for SUSE Linux Enterprise Server 10 is /bin/bash.
- **ShadowMax: 99999.** This attribute sets the number of days before the password expires.
- **ShadowWarning: 7.** Users can be warned before their passwords expire. This attribute sets the number of days before the warning is issued. Set to -1 to disable the warning.
- **ShadowInactive: -1.** This attribute sets the number of days that a user can still log in after the password expires. Set to -1 to set an unlimited number of days.
- **ShadowMin: 0.** This attribute sets the minimum number of days that need to pass before a password can be changed.
- **ShadowLastChange: 12609.** This attribute sets the date of the last password change.

The home directories of the users are normally stored on a server. You use automounter to mount the user's home directory.

Objective 4 Query Information from the LDAP Server

You can use the command **ldapsearch** to read data from the LDAP directory. The following command reads the entire tree:

ldapsearch -x

The **-x** option forces ldapsearch to use the simple authentication method. This is necessary if the LDAP server is not yet configured to use the SASL authentication method.

ldapsearch reads the search base for the query out of the configuration file /etc/ldap.conf. The search base is the entry in the directory where ldapsearch starts the recursive search process.

If the file ldap.conf file does not exist, or if you want to use a different search base, you can specify it with the **-b** option, as in the following:

ldapsearch -x -b "ou=people,dc=digitalairlines,dc=com"

If you have a lot of data in your LDAP tree, you might want to limit the output of ldapsearch to specific entries. You can do that by adding a filter expression to the ldapsearch command, as in the following:

ldapsearch -x "(uid=g*)"

In this example, ldapsearch displays all entries that have a uid attribute starting with **g**. You can use any attributes or objectClasses as a search filter.

The output of ldapsearch looks like the following:

```
da51:/etc/openldap # ldapsearch -x "uid=geeko"
# extended LDIF
#
# LDAPv3
# base <> with scope subtree
# filter: (uid=geeko)
# requesting: ALL
#

# geeko, people, digitalairlines, com
dn: uid=geeko,ou=people,dc=digitalairlines,dc=com
objectClass: posixAccount
objectClass: shadowAccount
objectClass: inetOrgPerson
uid: geeko
uidNumber: 1000
gidNumber: 100
cn: Geeko Chameleon
givenName: Geeko
sn: Chameleon
homeDirectory: /home/geeko
loginShell: /bin/bash
shadowMax: 99999
shadowWarning: 7
shadowInactive: -1
shadowMin: 0
shadowLastChange: 12609

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
da51:/etc/openldap #
```

ldapsearch displays the result in LDIF format. That means you can transfer the data to another LDAP server by redirecting the data into a file and loading it with ldapadd on a different machine.

Objective 5 Modify and Delete Entries of the LDAP Server

The easiest way to modify data in the LDAP directory in SUSE Linux Enterprise Server 10 is to modify an LDIF file and apply the changes with the **ldapmodify** tool.

In the following example, the uidNumber of the user geeko has been changed to 1011:

```
dn: uid=geeko,ou=people,dc=digitalairlines,dc=com
objectClass: posixAccount
objectClass: shadowAccount
objectClass: inetOrgPerson
uid: geeko
uidNumber: 1011
gidNumber: 100
cn: Geeko Chameleon
givenName: Geeko
sn: Chameleon
homeDirectory: /home/geeko
loginShell: /bin/bash
shadowMax: 99999
shadowWarning: 7
shadowInactive: -1
shadowMin: 0
shadowLastChange: 12609
```

To apply the changes, use the following command:

ldapmodify -x -D "cn=Administrator,dc=digitalairlines,dc=com" -W -f geeko.ldif

Using the ldapmodify command is similar to using ldapadd. ldapmodify compares the data in the directory with the data from the LDIF file and applies the changes to the directory entries.

To delete an entry from the LDAP directory, use the following command:

ldapdelete -D "cn=Administrator,dc=digitalirlines,dc=com" -x -W "uid=geeko,ou=people,dc=digitalairlines,dc=com"

In this example, the entry with the distinguished name "uid=geeko,ou=people,dc=digitalairlines,dc=com" is deleted.

Exercise 3-2 Add Users to the LDAP Directory

In this exercise, you add a user to an LDAP directory.

You will find this exercise in the workbook.

(End of Exercise)

Objective 6 Activate LDAP Authentication

- Change the User Password
- Activate pam_ldap

Change the User Password

If you use the LDAP directory for user authentication, you can change the user's password by using the **ldappasswd** command.

```
da51:~ # ldappasswd -x -D
"cn=Administrator,dc=digitalairlines,dc=com" -W -S
"uid=geeko,ou=people,dc=digitalairlines,dc=com"
New password:
Re-enter new password:
Enter LDAP password:
Result: Success (0)
da51:~ #
```

First you have to enter the new password for the user twice. Then you have to enter the administrator's password to fulfill the change.

The user can change his or her own password by:

```
da51:~ # ldappasswd -x -D
"uid=geeko,ou=people,dc=digitalairlines,dc=com" -W -S
"uid=geeko,ou=people,dc=digitalairlines,dc=com"
New password:
Re-enter new password:
Enter LDAP password:
Result: Success (0)
da51:~ #
```

If you want to create a home directory automatically when the user logs in the first time, add the following line to /etc/pam.d/sshd
session required pam_mkhomedir.so skel=/etc/skel/ umask=0022

If the user now searches for the new user's LDAP entry he should be prompted to enter his password and the crypted password should be shown in the search results.

```
da51:~ # ldapsearch -x -D
"uid=geeko,ou=people,dc=digitalairlines,dc=com" -W
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <> with scope subtree
# filter: uid=geeko
# requesting: ALL
#

# geeko, people, digitalairlines.com
dn: uid=geeko,ou=people,dc=digitalairlines,dc=com
objectClass: posixAccount
objectClass: shadowAccount
objectClass: inetOrgPerson
uid: geeko
uidNumber: 1000
gidNumber: 100
cn: Geeko Chameleon
givenName: Geeko
sn: Chameleon
homeDirectory: /home/geeko
loginShell: /bin/bash
shadowMax: 99999
shadowWarning: 7
shadowInactive: -1
shadowMin: 0
shadowLastChange: 12609
userPassword::
e1NTSEF9TVJjWDlHQm8ydGxIUE1HWWFUT2lOQWZlNDkrUHA4OTU=

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
da51:~ #
```

Activate pam_ldap

To activate PAM and NSS, you have to add LDAP as a possible source for authentication. You have to make sure the following lines are in file /etc/nsswitch.conf:

```
passwd: compat
group:  compat
...
passwd_compat:  ldap
group_compat:   ldap
```

If you configure the LDAP client with YaST the PAM module pam_ldap.so is activated for all kinds of authentication. This is done in the file /etc/security/pam_unix2.conf.

```
auth:   use_ldap
account:        use_ldap
password:       use_ldap
session:        none
```

If you configure the LDAP client manually, you can activate the LDAP authentication for selected services.

To do this, use the PAM module pam_ldap.so (included in the package pam_ldap) is used. Add the module corresponding to the service to the PAM configuration file in /etc/pam.d.

For example, if you want LDAP authentication for ssh, the file /etc/pam.d/sshd must must look like this:

```
#%PAM-1.0
auth     include        common-auth
auth     required       pam_nologin.so
account  include        common-account
password include        common-password
session  include        common-session
# Enable the following line to get resmgr support for
# ssh sessions (see /usr/share/doc/packages/resmgr/README)
#session  optional      pam_resmgr.so fake_ttyname
```


Here the files common-auth, common-account, common-password and common-session are included. These files can also be found in /etc/pam.d/. The file /etc/pam.d/common-auth has -for example- the following statements:

```
auth    required        pam_env.so
auth    required        pam_unix2.so
```

If a required module fails, the modules following the failed module are tested anyway. But an error occurs at the end. If a sufficient module fails, but another sufficient module matches, login is allowed.

There are a couple of examples available in the directory /usr/share/doc/packages/pam_ldap/pam.d/.

The advantage of using LDAP for all kinds of authentication (as done by YaST) is that you can use standard tools like **passwd** or **chsh**.

```
kbailey@da51:~> passwd
Changing password for kbailey.
Enter login(LDAP) password:
New password:
Re-enter new password:
LDAP password information changed for geeko
```

Exercise 3-3 Set up an LDAP User Database

In this exercise, you set up an LDAP user database.

You will find this exercise in the workbook.

(End of Exercise)

Objective 7 Replicate OpenLDAP Servers

If there are a lot of requests for the LDAP server you may need more than one server. One server is the master. The other servers are slaves. A master/slave arrangement provides an effective way to increase capacity, availability, and reliability.

Changes on the master server are propagated by the slurpd to the slaves. In this objective we cover the following topics:

- Add the Replicaton DN to the LDAP Directory
- Configure slapd for Replication
- The Command-Line Options of slurpd
- Transfer the LDAP Database

Add the Replicaton DN to the LDAP Directory

In the LDAP directory (on master and slave) there must be an object with the DN used for replication. This can look like this:

```
dn: uid=replicator,dc=digitalairlines,dc=com
objectClass: inetOrgPerson
uid: replicator
cn: LDAP Replicator
sn: Replicator
```

You also have to set a password for this object using **ldappasswd**.

Configure slapd for Replication

The slapd on the master server can be configured to write a replication logfile by the following entry /etc/openldap/slapd.conf:

```
replogfile /var/lib/ldap/master-slapd.replog
```


This file contains regular LDIF change records, e.g.

```
time: 1148659780
dn: ou=groups,dc=digitalairlines,dc=com
changetype: add
objectClass: organizationalUnit
ou: groups
structuralObjectClass: organizationalUnit
entryUUID: c7b0c4e0-811d-102a-9c4d-2129ebe1b381
creatorsName: cn=Administrator,dc=digitalairlines,dc=com
createTimestamp: 20060526160940Z
entryCSN: 20060526160940Z#000000#00#000000
modifiersName: cn=Administrator,dc=digitalairlines,dc=com
modifyTimestamp: 20060526160940Z
```

On the master server you have to specify the name(s) of the slave server(s) with the **replica** option:

```
replica uri=ldap://10.0.0.51:389
binddn="cn=replicator,dc=digitalairlines,dc=com" bindmethod=simple
credentials=novell
```

The following parameters are used:

- **uri**. Name or IP of the slave server. The default LDAP port is 389. If you want to use LDAPS (port 636), the uri looks like this
  replica uri=ldaps://10.0.0.51:636 ...
- **binddn**. DN of the master server that is allowed to access the slave. This DN must also be specified on the slave server with the option **updatedn**.

> This DN should not be the same as the master's rootdn.

- **bindmethod**. Specifies the authentication method. You can select between **simple** (password-based authentication) and **sasl** (SASL authentication).

- **credentials**. Password for simple authentication.

On the slave server you have to add the **updatedn** option, as mentioned. For example:

```
updatedn=”cn=replicator,dc=digitalairlines,dc=com”
```

Use the **updateref** directive to define the URL the slave should return if an update request is received.

```
updateref="ldap://10.0.0.51"
```

You have to make sure that this DN has permission to write the database (e.g., by **access** directives).

```
access to *
  by dn="cn=replicator,dc=digitalairlines,dc=com" write
  by * read
```

The Command-Line Options of slurpd

The man page of slurpd can be accessed by **man 8 slurpd**.

The options of **slurpd** are:

- **-d** ***level***. Specify the debug level. Possible debug levels are:
    - **4**. Heavy trace debugging
    - **64**. Configuration file processing
    - **65535**. Enable all debugging
    - **?**. Lists als debug levels and exits slurpd.
- **-f** ***file***. Specifies the slapd configuration file. The default is /etc/openldap/slapd.conf.
- **-r** ***file***. Specifies the name of the slapd replication logfile.

- **-o**. Run in "one-shot" mode. Normally, slurpd processes the replog file and then watches for more replication entries to be appended. In one-shot mode, slurpd processes a replication log and exits.
- **-t *directory***. slurpd copies the replication log to a working directory before processing it.

There is a start script available in /etc/init.d/ and you can start the slurpd by entering **rcslurpd start**.

Transfer the LDAP Database

Before you start the slapd on the slave server, you have to transfer the LDAP database from the master to the server.

Make sure that the slapd of the master is shut down. To transfer the database (/var/lib/ldap/ by default), you can use **scp**.

Alternatively you can use the command **slapcat**. **slapcat** is used to generate an LDIF output based upon the contents of a slapd database. By default the output is written to STDOUT. You can specify a file by using the option **-l**.

slapcat has other interesting options as well (e.g., for filtering). See the man page for further details (**man 8 slapcat**).

Exercise 3-4 Replicate OpenLDAP Servers

In this exercise, you configure LDAP replication with your neighbor.

You will find this exercise in the workbook.

(End of Exercise)

Summary

Objective	Summary
1. Understand the Basics of LDAP	Directory services are tree-like structured databases that contain entry-based information. OpenLDAP is the most popular open source LDAP directory and can be used for user authentication in SUSE Linux Enterprise Server 10.
2. Install and Set Up an OpenLDAP Server	If you did not configure an OpenLDAP server during the installation, you need to install the following software packages: ■ openldap2 ■ openldap2-client The configuration of the OpenLDAP server is located in the file /etc/openldap/slapd.conf. You can create passwords for the administrator entry of the configuration file with the command **slappasswd**. /etc/openldap/ldap.conf is the configuration file of the LDAP client. /etc/ldap.conf is the configuration file for the LDAP nameservice switch library, the LDAP PAM module, and the shadow package.

Objective	Summary
3. Add Entries to the LDAP Server	Use **ldapadd** to insert data from LDIF files into the directory. Make sure that LDIF files conform to Unicode.
4. Query Information from the LDAP Server	Use **ldapsearch** to query information from the directory.
5. Modify and Delete Entries of the LDAP Server	Use **ldapmodify** to change entries in the directory. Use **ldapdelete** to delete directory entries.
6. Activate LDAP Authentication	If you use the LDAP directory for user authentication, you can change the user's password by using the **ldappasswd** command. If you configure the LDAP client with the YaST module, the PAM module pam_ldap.so is activated for all kinds of authentication in the file /etc/security/pam_unix2.conf. If you configure the LDAP client manually, you can activate the LDAP authentication only for selected services. To do this, use the PAM module pam_ldap.so (included in the package pam_ldap) is used. Add the module corresponding to the service to the PAM configuration file in /etc/pam.d.

Objective	Summary
7. Replicate OpenLDAP Servers	The slapd on the master server can be configured to write a replication logfile. On the master server you also have to specify the name(s) of the slave server. On the master you specify a DN to used by the master to access the slave. This DN must also be specified on the slave server and must have permission to write to the database. **slapcat** is used to generate an LDIF output based on the contents of a slapd database.

SECTION 4 Configure a Mail Server

The standard mail server in SUSE Linux Enterprise Server 10 is Postfix.

Postfix was written as an alternative to the well-known Mail Transfer Agent (MTA) Sendmail.

Postfix was originally developed as VMailer by Wietse Venema during his employment at IBM and placed under the “IBM Public License Version 1.0 – Secure Mailer.”

Before its publication, however, the program was renamed Postfix to exclude possible name conflicts.

The “IBM Public License Version 1.0 - Secure Mailer” is, to a large degree, similar to the GNU GPL. It allows free distribution of the source code and binary package of the program and allows modifications to the program on condition that the license is always included.

Venema had several goals for Postfix:

- It should be a fast mailer.
- It should be easy to administer.
- It should be secure.
- It should be compatible with Sendmail.

Objectives

1. Understand the Function of the Three Mail Agents
2. Use the Basic Linux mail Command
3. Understand the Architecture and Components of Postfix
4. Start, Stop, and Reinitialize Postfix
5. Configure Postfix
6. Use Postfix Tools
7. Receive Email over IMAP and POP3
8. Manage Spam
9. Use a Virus Scanner for Email

Objective 1 Understand the Function of the Three Mail Agents

Programs that process Internet messages are called ***agents***.

There are three types of agents:

- Mail Transfer Agent (MTA)
- Mail Delivery Agent (MDA)
- Mail User Agent (MUA)

The following is also described:

- The Mail Cycle
- Simple Mail Transfer Protocol (SMTP)

Mail Transfer Agent (MTA)

Emails are passed by the email software to the MTA.

The MTA sends them to a recipient MTA, which checks if the destination address exists locally. If it does, the recipient MTA passes the mail to an MDA.

If the address does not exist locally, the recipient MTA sends the message back to the sending MTA.

Chains are also possible: an MTA can send a message to another MTA, which passes it to other MTAs.

Well-known MTAs are

- Sendmail
- Postfix
- Exim
- Qmail

Each recipient MTA leaves a **received** entry in the header of the message sent.

Mail Delivery Agent (MDA)

The MTA specifies which local user the message is intended for and passes it to an MDA, which stores it in the right location.

A well-known MDA is the program Procmail.

Mail User Agent (MUA)

The MUA reads saved messages and passes new messages to the MTA. It is the interface between the MDA and the user.

The MUA can receive saved messages in three different ways:

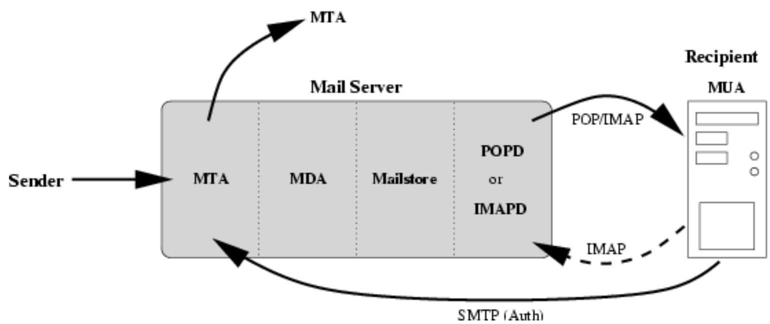
- By mail access protocols (such as IMAP or POP)
- Remotely by file access protocols
- Through access to local files

When the MUA goes through IMAP or POP, its functionality can be separated into the mail client and a server it belongs to, located between the client and the storage location.

The Mail Cycle

The following figure shows the mail cycle:

Figure 4-1



When an MTA receives an email, it checks first if the recipient of the email is an existing user on the local machine. If the email is not for a local user, the MTA sends it to another MTA (either a so-called relay host or to the mail server responsible for the destination domain).

If the email is for a local user, the MTA hands it over to the MDA which stores the mail in the mailbox of the respective user. From there, the email can be retrieved by using POP or IMAP.

If POP is used, the email is transferred to the client computer; additionally it can be kept on the server.

If IMAP is used, all emails are kept on the server and the MUA only checks emails on this server. The emails are not transferred to the client computer.

When sending email, the MTA can be configured to ask the MUA for authentication of the user. This can be used to prohibit sending spam emails via this MTA.

Simple Mail Transfer Protocol (SMTP)

Sending email is controlled uniformly in the Internet by SMTP (Simple Mail Transfer Protocol). SMTP is defined in RFC 821. Since its creation in 1982, it has undergone numerous extensions, such as Mime format (RFC 2045) and SMTP Service Extensions (RFC 1869).

SMTP has the following characteristics:

- There is a direct TCP connection between the sending and receiving computers - between the processes involved.
- The task of SMTP is only transmission, not the receipt or saving of messages.
- Every message is confirmed.
- Messages can be forwarded, for example, if an address has changed.
- Client and server communicate through readable plain text commands.

In the following we want to have a closer look at:

- The SMTP Commands
- Command Syntax
- SMTP Reply Codes
- Minimal SMTP Command Implementation
- An Example for Sending Mail with Telnet

The SMTP Commands

The commands are:

- **HELO** (Hello) is used by the SMTP sender to open a connection to an SMTP recipient (a greeting) and to introduce itself with its full host name. This host name is passed as an argument attached to the **HELO** command.

  The SMTP recipient answers this greeting with its full host name. This completes the initialization of communication between the SMTP sender and the recipient.

- **MAIL FROM** initializes transmission of an email message. In the simplest case, the **MAIL FROM** command takes the sender's email address (reverse-path) as an argument. The argument can be optionally extended to include a list of hosts in front of the recipient's address via which a reply should be routed (source route relaying), e.g.:
  - @venus.example.com
  - @mars.example.com:Sales@example.com

- **RCPT TO** (Recipient) sets the recipient's address (forward-path) for an item of mail. If the mail should be sent to several recipients at the same time, this command is repeated that number of times.

  The command may also include a list of hosts (e.g., @mars.example.com,@venus.example.com:info@example.com) via which the mail should be routed (source route relaying).

  If the argument contains source route relaying through a list of hosts, the mail will be routed directly to the first host given. This host removes itself from the beginning of the list and forwards the mail to the next host in the list.

- **DATA** tells the SMTP mailer that anything that follows is the content of the mail. The end of the mail content is indicated by Enter.

When the SMTP recipient recognizes the end of the mail transmission, it begins to process the information received. This involves interpreting and possibly updating the forward and reverse paths. If this data processing results in no failures, the SMTP sender is sent an **OK**. Otherwise, an error message is sent.

A time stamp is inserted at the beginning of the mail as soon as the SMTP recipient receives a mail to forward to a further SMTP recipient or for direct delivery to the recipient. This time stamp contains information about the identity of the SMTP sender, the SMTP recipient, and the time when the mail reached the SMTP recipient.

Mails that are routed over several SMTP relays contain a corresponding number of these time stamps. If the SMTP recipient is responsible for the ultimate delivery of the mail (if, for example, the addressee's mailbox is located on the SMTP recipient), the current contents of the return path (i.e., the contents of the **MAIL FROM** argument after being modified through the previous SMTP relays) are inserted at the top of the mail.

- **VRFY** (Verify) verifies a user ID. This causes the SMTP recipient to check the validity of the addressee given as an argument. If the SMTP reader knows the addressee, the address is expanded into a full address (including domain). The **VRFY** command has no influence on the **MAIL FROM**, **RCPT TO**, and **DATA** instructions.
- **EXPN** (Expand) instructs the SMTP recipient to treat the argument as a mailing list. As a result, the members of the list are transmitted to the SMTP sender.
- **RSET** (Reset) resets all the information previously stored about the SMTP recipient. The forward path, reverse path, and mail contents are lost.

- **HELP** takes as its argument any other SMTP command. A page of tips about how to use the corresponding instruction will be displayed.
- **NOOP** (No Operation) causes the SMTP recipient to answer with an **OK**. Apart from that, no changes are made to the mail content or the forward and return paths.
- **QUIT** ends the connection between the SMTP sender and recipient.

Command Syntax

SMTP commands provide direct communication between the sender and recipient. The most commonly used SMTP commands are described in the following table.

Table 4-1

Command Syntax	Description of the Arguments
HELO *hostname*	***hostname*** contains the complete host name of the SMTP sender.
MAIL FROM: *reverse-path*	***reverse-path*** contains the relay path for source route relaying and the sender's address.
RCPT TO: *forward-path*	***forward-path*** contains the relay path for source route relaying and the recipient's address.
DATA ***data*** **.**	***data*** constitutes the contents of the mail (the actual message).
RSET	./.
VRFY *string*	***string*** contains a user or mailbox name.
EXPN *string*	***string*** contains a mailing list identifier.
HELP [*string*]	***string*** contains any SMTP command.

Table 4-1

Command Syntax	Description of the Arguments
NOOP	./.
QUIT	./.

Commands can be written in lower case or upper case; SMTP commands are not case sensitive. For example, the SMTP recipient will treat the following commands in the same way: **MAIL FROM**, **Mail From**, **mail from**, **MaIl FrOm**.

In contrast, the arguments in ***forward-path*** and ***reverse-path*** may be case-sensitive. The interpretation of these arguments depends on the SMTP recipient's operating system and the structure of the user database, which is why lowercase and uppercase characters may be treated differently.

SMTP Reply Codes

During communication between the SMTP sender and SMTP recipient, the sender transmits various commands to the recipient and controls the course of the communication as a whole. The recipient acknowledges the message's receipt and gives the status of command processing with a corresponding reply code. Only when the sender has received a receipt for the previous command can it start transmitting the next command.

The most commonly found SMTP reply codes are listed in the table below:

Table 4-2

Reply Code	Description
211	System status or system help reply.
214	Displays the help message after entering **HELP**.
220	SMTP server ready.

Table 4-2 *(continued)*

Reply Code	Description
221	SMTP server has closed connection.
250	Specified command has been carried out.
251	User not local; will forward to **forward-path**.
354	Start of the DATA entry.
421	SMTP service not available.
450	Requested action not taken (mailbox is already in use).
451	Requested action aborted.
452	Requested action not taken; insufficient storage space in system.
500	Syntax error, command unrecognized.
501	Syntax error in parameters or arguments.
502	Command not implemented.
503	Bad sequence of commands.
504	Command not implemented for that parameter.
550	Requested action not taken; file unavailable (e.g., mailbox not found, no access).
551	User does not exist on mail server; try ***forward-path***.
552	Requested mail action aborted: storage allocation exceeded.
553	Requested action not taken: mailbox name not allowed (e.g., mailbox syntax incorrect).
554	Transaction failed.

Minimal SMTP Command Implementation

Not all commands introduced up to now are implemented by every server. Certain SMTP commands are intentionally not included in some server implementations to increase the security of the server. One such security-critical command is **HELP**. The Postfix SMTP server by Wietse Venema has not fully implemented this command and simply responds to a **HELP** command with reply code 502 ("Command not implemented").

The following list shows the SMTP commands that an SMTP server absolutely must implement to provide SMTP communication:

- **HELO**
- **MAIL FROM:**
- **RCPT TO:**
- **DATA**
- **RSET**
- **NOOP**
- **QUIT**

An Example for Sending Mail with Telnet

The text below shows a typical example of the log output that can easily be reconstructed using a telnet session on port 25 (standard mail server port):

```
da53:~ # telnet da51.digitalairlines.com 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 da51.digitalairlines.com ESMTP Postfix
HELO da53.digitalairlines.com
250 da51.digitalairlines.com
MAIL FROM: jgoldman@digitalairlines.com
250 Ok
RCPT TO: geeko@digitalairlines.com
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
sed diem nonummy nibh euismodtincidunt ut lacreet dolore
magna aliguam erat volutpat. Ut wisis enim ad minim veniam,
quis nostrud exerci tution ullamcorper suscipit lobortis
isl ut aliquip ex ea commodo consequat.
.
250 Ok: queued as 45B5116A9D
QUIT
221 Bye
Connection closed by foreign host.
da53:~ #
```

Objective 2 Use the Basic Linux mail Command

You can use the command mail in one of two ways:

- Read Status Mail
- Use the Simple Mail Client

Read Status Mail

Some programs send status mail messages or notes to the user root. If you log in at a virtual terminal, you get a notification if there are mail messages for you:

```
da51 login: root
Password:
You have new mail in /var/mail/root.
Last login: FRI Jul 18 13:23:51 from 10.0.0.6
da51:~ #
```

To read these messages, you can use the **mail** command:

```
da51:~ # mail
mailx version nail 10.6 11/15/05. Type ? for help.
"/var/mail/root": 3 messages 3 new
>N 1 root@da51.digital Wed Jul 9 15:02 28/894  SuSEconfig:
xntp.caveats
 N 2 root@da51.digital Wed Jul 9 15:02 43/1836 SuSEconfig:
SuSEfirewall2
 N 3 root@da51.digital Wed Jul 9 15:02 151/5888 SuSEconfig:
openssh-chang
?
```

The prompt of mail is a “?”. At the prompt, you can use the following commands:

Table 4-3

Command	Purpose
t *message_list*	Type messages.
n	Go to and type the next message.
e *message_list*	Edit messages.
f *message_list*	Give headlines of messages.
d *message_list*	Delete messages.
u *message_list*	Undelete messages.
R *message_list*	Reply to message senders.
r *message_list*	Reply to message senders and all recipients.
m *user_list*	Mail to specific users.
q	Quit, saving unresolved messages in mbox.
x	Quit, do not remove system mailbox.

A *message_list* consists of integers, ranges of integers, or user names separated by spaces. If omitted, mail uses the last message typed.

A *user_list* consists of user names or aliases separated by spaces. Aliases are defined in .mailrc in your home directory.

Use the Simple Mail Client

You can use the program mail as a normal mail client on the command line. The simplest form of using mail is **mail *mail_address***.

Instead of *mail_address* you can enter a username of a person inside your local network. (For sending messages to users other than the local users, you need to configure a mail server on your local host.)

mail first asks you for the subject of the mail. After this you can enter the body of your message. To finish the text you have to enter a single full stop in the last line:

```
geeko@da51:~ > mail root
Subject: A simple Mail
This is just a test mail.
The body of the text ends with an single full stop.
.
```

mail confirms the end of the text body with “EOT” (end of text) and sends the mail immediately.

The following are the most important options of mail:

Table 4-4

Option	Description
-a *file*	Attach the given *file* to the message.
-b *list*	Send blind carbon copies to *list*. List should be a comma-separated list of names.
-c *list*	Send carbon copies to *list* of users.
-q *file*	Start the message with the contents of the specified *file*. Can be given in send mode only.
-s *subject*	Specify *subject* on command line (only the first argument after the -s flag is used as a subject; be careful to quote subjects containing spaces).

Table 4-4 *(continued)*

Option	Description
-R *address*	Specify reply-to *address* on command line. Only the first argument after the -R flag is used as the address.
-F	Read emails are stored in the file ~/mbox. The -F mail option opens this file to read these old email messages.

Exercise 4-1 Send Mail to root

In this exercise, you send an email to user root using the **mail** command. You also switch to user root and read this mail.

You will find this exercise in the workbook.

(End of Exercise)

Objective 3 Understand the Architecture and Components of Postfix

Venema met his Postfix design goals using a series of modular function units.

Unlike Sendmail, Postfix is not a large monolithic program block. Instead, it consists of a variety of small programs, each of which is allocated a specific task (for example, accepting an email).

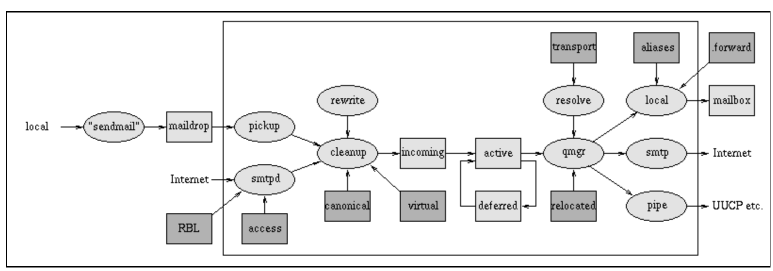
This modularization makes the system more transparent.

The individual components are easier to administer, facilitating further development of Postfix.

The following figure, taken from the original Postfix documentation, shows a rough summary of the modularization of Postfix.

Modules that are not covered at this stage are in /usr/share/doc/packages/postfix/html/OVERVIEW.html.

Figure 4-2



Individual Postfix processes are represented in the diagram by ellipses. Dark squares stand for lookup tables and light squares represent mail queues or mailboxes.

For security reasons, Postfix works with four mail queues. For every mail queue, there is a directory bearing the same name under /var/spool/postfix/.

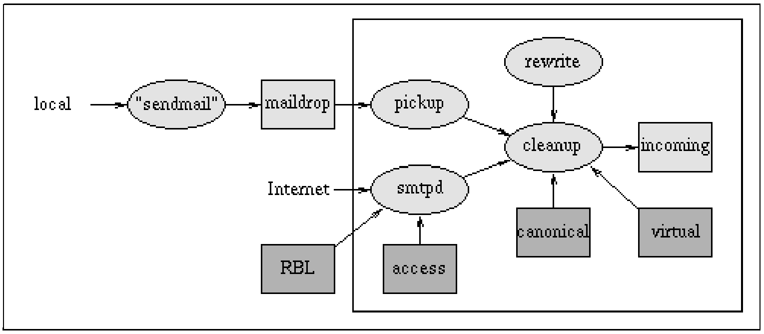
The functions of the queues and the Postfix files are described in

- Process of Inbound Email
- Process of Outbound Email
- Components of the Postfix Program Package

Process of Inbound Email

The following figure shows how an email can reach Postfix and how it is processed.

Figure 4-3



The following describe these processes:

- Email Received Locally
- Email Received over the Network

Email Received Locally

Postfix uses the postdrop command to place an email sent locally into the maildrop queue before it is picked up by the pickup daemon.

The pickup daemon checks it for content, size, and other factors based on rules; then it passes the email to the cleanup daemon.

The cleanup daemon does the following:

- Inserts missing header lines (Resent:, From:, To:, Message-ID:, Date:) in the email (if the mail was written with telnet)
- Deletes double recipient addresses
- Uses the trivial-rewrite daemon (/usr/lib/postfix/trivial-rewrite) to convert the email address in the header to the ***user@fully-qualified-domain*** convention
- Writes data in the header according to the rules in the lookup tables /etc/postfix/canonical and /etc/postfix/virtual

After this, the email is copied to the incoming queue and the queue manager /usr/lib/postfix/qmgr is informed of the arrival of this email.

Email Received over the Network

Email received over the Internet or LAN is accepted by the daemon, smtpd. smtpd checks the email for content, size, and other factors before passing it to the cleanup daemon.

The cleanup daemon does the following:

- Replaces missing header lines (Resent:, From:, To:, Message-ID:, Date:) in the email
- Deletes double recipient addresses

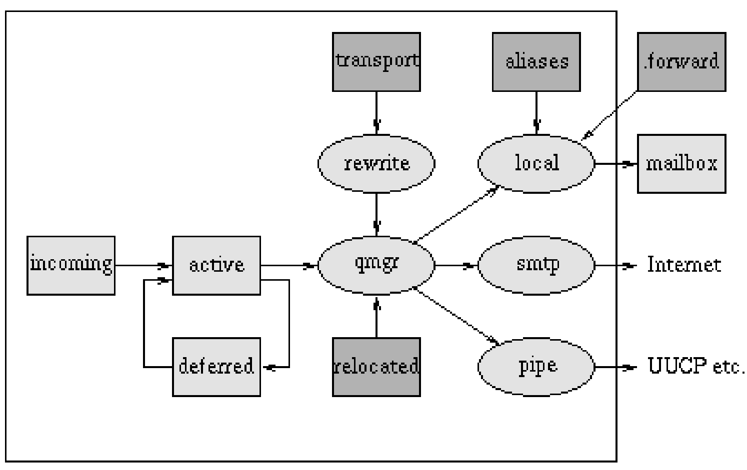
- Uses the trivial-rewrite (/usr/lib/postfix/trivial-rewrite) daemon to convert the email address in the header to the ***user@fully-qualified-domain*** convention
- Writes data in the header according to the rules of the lookup tables /etc/postfix/canonical and /etc/postfix/virtual

Then the email is copied to the incoming queue and the queue manager /usr/lib/postfix/qmgr is informed of the arrival of this email.

Process of Outbound Email

The following figure shows how an email is handled byPostfix before it leaves the system to be delivered:

Figure 4-4



The following topics describe this process:

- Deliver Email to Local Users
- Deliver Email to Users on Remote Systems
- Process Undeliverable Emails

Deliver Email to Local Users

The queue manager fetches an email from the incoming queue and copies it to the active queue as soon as the active queue contains no other emails.

The trivial-rewrite daemon takes over the checking procedure based on the lookup table /etc/postfix/transport to see whether the recipient of the email is on the local system or a remote system.

If this daemon decides the email should be delivered locally, the queue manager orders the local delivery service (/usr/lib/postfix/local) to deliver the email to the recipient's mailbox, taking into account the alias database (/etc/aliases) as well as any forward files of the user (~/.forward).

The local daemon can also be configured to have mail delivered by external programs, such as Procmail.

Deliver Email to Users on Remote Systems

The queue manager fetches an email from the incoming queue and copies it to the active queue, as soon as the active queue is empty.

The trivial-rewrite daemon uses the /etc/postfix/transport lookup table to see if the recipient of the email is on the local system or on a remote system.

If the daemon decides the email should be delivered to a remote system, the queue manager activates the SMTP service to deliver the email.

The SMTP service tries to find the mail exchanger specified for the target host; then it delivers the email to the mail exchanger for the recipient host.

Process Undeliverable Emails

Emails that cannot be delivered are removed from the active queue by the queue manager and copied to the deferred queue.

The queue manager then copies this email at regular intervals from the deferred queue back to the active queue and tries again to deliver the email.

Components of the Postfix Program Package

During the Postfix installation, files are saved to various locations on a SUSE Linux Enterprise Server 10 system. These locations can be grouped according to the following criteria:

- **/etc/aliases**. This is the only file in /etc/. It has the same format as the aliases file for the MTA Sendmail and contains local address aliases.
- **/etc/postfix/**. All the configuration files defining Postfix mail processing are in this directory.

  Normally, the Postfix administrator is the only one who can make changes to these files.
- **/usr/lib/postfix/**. This directory contains all the programs needed directly by Postfix. To be more precise, these are the Postfix binaries.

These programs are not accessed directly by the system administrator.

- **/usr/sbin/**. This directory contains the administration programs for maintaining and manually controlling Postfix.

  An administrator uses these programs during maintenance work.

- **/usr/bin/**. This directory contains symbolic links with the names mailq and newaliases.

  Both links point to the program /usr/sbin/sendmail that provides a Sendmail-compatible administration interface for Postfix.

- **/var/spool/postfix/**. This directory contains the queue directories for Postfix and the directories etc/ and lib/ for Postfix processes that run in a chroot environment.

  If the variables POSTFIX_CHROOT and POSTFIX_UPDATE_CHROOT_JAIL in /etc/sysconfig/postfix are set to **yes**, these two directories are set up by

  SuSEconfig --module postfix

- **/usr/share/man/man[1|5|8]/**. These directories contain the manual pages for the Postfix binaries, for the configuration files, and the administration programs.

- **/usr/share/doc/packages/postfix/**. This contains documentation for Postfix.

The subdirectory html/ contains a detailed HTML description of Postfix and a very useful FAQ.

Objective 4 Start, Stop, and Reinitialize Postfix

Postfix runs by default on SUSE Linux Enterprise Server 10 and it is started in runlevel 3 during startup.

To start Postfix manually, you can use the command

rcpostfix start

To stop Postfix, enter

rcpostfix stop

To check the status of Postfix, enter

rcpostfix status

This command provides information on whether Postfix has already been activated on this system.

If the Postfix configuration has been changed, Postfix has to re-read the modified configuration files to adjust mail processing to the changed configuration.

To do this, enter

rcpostfix reload

or

rcpostfix restart

Objective 5 Configure Postfix

This objective covers the following topics:

- Configure the Postfix Master Daemon
- Configure Global Settings
- Configure General Scenarios
- Configure the Lookup Tables

Configure the Postfix Master Daemon

The Postfix master daemon /usr/lib/postfix/master is started directly by Postfix when the system is booted and is terminated only when the system goes down or if Postfix ends.

The Postfix master daemon is normally configured once only when as the email system is set up, and is usually never changed.

The master daemon, which monitors the entire mail system,

- Controls and monitors individual Postfix processes.
- Adheres to configured resource limits, which were defined in the file master.cf.
- Restarts killed Postfix processes.

The Postfix master daemon is configured in the file /etc/postfix/master.cf. Each line in the file contains an entry for one Postfix process.

The behavior of each process is defined by the configuration in the respective line:

```
#
==========================================================================
=
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (100)
#
==========================================================================
=
smtp      inet  n       -       n       -       -       smtpd
#smtps    inet  n       -       n       -       -       smtpd
#  -o smtpd_tls_wrappermode=yes -o smtpd_sasl_auth_enable=yes
#submission   inet    n       -       n       -       -       smtpd
#  -o smtpd_enforce_tls=yes -o smtpd_sasl_auth_enable=yes -o
smtpd_etrn_restrict
ions=reject
#628      inet  n       -       n       -       -       qmqpd
pickup    fifo  n       -       n       60      1       pickup
cleanup   unix  n       -       n       -       0       cleanup
qmgr      fifo  n       -       n       300     1       qmgr
#qmgr     fifo  n       -       n       300     1       oqmgr
#tlsmgr   fifo  -       -       n       300     1       tlsmgr
rewrite   unix  -       -       n       -       -       trivial-rewrite
bounce    unix  -       -       n       -       0       bounce
defer     unix  -       -       n       -       0       bounce
trace     unix  -       -       n       -       0       bounce
verify    unix  -       -       n       -       1       verify
flush     unix  n       -       n       1000?   0       flush
proxymap  unix  -       -       n       -       -       proxymap
smtp      unix  -       -       n       -       -       smtp
relay     unix  -       -       n       -       -       smtp
#       -o smtp_helo_timeout=5 -o smtp_connect_timeout=5
showq     unix  n       -       n       -       -       showq
error     unix  -       -       n       -       -       error
local     unix  -       n       n       -       -       local
virtual   unix  -       n       n       -       -       virtual
lmtp      unix  -       -       n       -       -       lmtp
anvil     unix  -       -       n       -       1       anvil
#localhost:10025 inet   n       -       n       -       -       smtpd -o
content
_filter=
...
```



```
...
#
# Interfaces to non-Postfix software. Be sure to examine the manual
# pages of the non-Postfix software to find out what options it wants.
#
# maildrop. See the Postfix MAILDROP_README file for details.
#
maildrop  unix  -       n       n       -       -       pipe
  flags=DRhu user=vmail argv=/usr/local/bin/maildrop -d ${recipient}
cyrus     unix  -       n       n       -       -       pipe
  user=cyrus argv=/usr/lib/cyrus/bin/deliver -e -r ${sender} -m
${extension} ${u
ser}
uucp      unix  -       n       n       -       -       pipe
  flags=Fqhu user=uucp argv=uux -r -n -z -a$sender - $nexthop!rmail
($recipient)
ifmail    unix  -       n       n       -       -       pipe
  flags=F user=ftn argv=/usr/lib/ifmail/ifmail -r $nexthop ($recipient)
bsmtp     unix  -       n       n       -       -       pipe
  flags=Fq. user=foo argv=/usr/local/sbin/bsmtp -f $sender $nexthop
$recipient
vscan     unix  -       n       n       -       10       pipe
  user=vscan argv=/usr/sbin/amavis ${sender} ${recipient}
procmail  unix  -       n       n       -       -       pipe
  flags=R user=nobody argv=/usr/bin/procmail -t -m /etc/procmailrc
${sender} ${r
ecipient}
```

If an entry in the file is too long for a specific service, this entry can be continued in the following lines by adding an empty space at the beginning of the following line; for example:

```
procmail  unix  -       n       n       -       -       pipe
  flags=R user=nobody argv=/usr/bin/procmail -t -m /etc/procmailrc
${sender} ${recipient}
```

The meaning of individual fields in a configuration line and their possible values are listed below.

Default values, if any, are listed in the description. If an entry is set to “–”, the default value is used.

- **service**. The name of the Postfix process.

  An entry for a service that is controlled by the inet daemon can be specified in the form ***host:port***.

  inet is the service that controls who can connect to your computer and which services they can use.

  An entry for the SMTP service could be

  localhost:smtp

  This entry would start the Postfix process /usr/lib/postfix/smtpd in such a way that it only receives email messages on port 25 of the loopback interface (if this port is entered correctly in the file /etc/services).

  The host prefix and the following colon are optional.

- **type**. Allows you to specify a connection type.

  Possible entries are

  - **inet** for Internet sockets (TCP/UDP)
  - **unix** for UNIX domain sockets (only for local communication)
  - **fifo** (first in, first out) for named pipes

- **private**. Configures access to the service.

  The value **y** (yes) only defines access to this service from the mail system.

  The entry **n** (no) also allows access to this service for components outside the mail system. For services of the type inet, the value **n** must always be set.

  The default value is **y**.

- **unpriv**. Configures the UID under which this service is running.

  With the value **y** (yes), the configured service runs under the unprivileged user configured in the file /etc/postfix/main.cf with the variable mail_owner (as a rule, the user postfix).

  If this value is set to **n** (no), the service runs with root privileges - with the UID 0.

  The default value is **y**.

- **chroot**. Specifies the chroot behavior of the service.

  The value **y** (yes) causes the service to be started in a chroot environment.

  The root path of this environment is defined in the variable queue_directory in the file /etc/postfix/main.cf (this is normally the directory /var/spool/postfix/).

  The default value is **y**.

- **wakeup**. Runs the service again after the given number of seconds have expired.

  The default value of **0** deactivates this function for the service.

  Currently only the pickup daemon and the queue manager use this function.

  The default value is **0** (never).

- **maxproc**. Defines the maximum number of processes that can be run simultaneously.

  The default value is defined in the variable default_process_limit in the file /etc/postfix/main.cf.

  The default value is **100**.

- **command + *args***. Configures the command to run, including the required arguments.

The path name of the command to run is relative to the directory defined in the file /etc/postfix/main.cf via the variable daemon_directory (this is normally the directory /usr/lib/postfix/).

If one or more **-v** arguments are given, the debugging level is increased for the given command.

Specifying the **-D** argument allows debugging by using the debugging command, specified in the file /etc/postfix/main.cf by the variable debugger_command.

Configure Global Settings

All other configuration definitions (apart from the configuration of processing rules in lookup tables) are set in the following file:

/etc/postfix/main.cf

On SUSE Linux Enterprise Server 10, the most common parameters of this file can be modified using variables in the files

- **/etc/sysconfig/mail**

  and

- **/etc/sysconfig/postfix**

Postfix is one of the last services that needs **SuSEconfig** to run for generation of the actual configuration files from files located in /etc/sysconfig/.

The file /etc/sysconfig/mail is used for general configurations that are not specific for Postfix and also used for Sendmail: For the MTA to operate correctly, you have to do the following in the file /etc/sysconfig/mail:

1. The fully qualified domain name (FQDN) must be entered in the variable FROM_HEADER.

   If this variable is not set, the host name (FQDN) will be used.

2. The variable SMTPD_LISTEN_REMOTE should be set to **yes** and Postfix will listen on port 25 for arriving mails.

   Otherwise, only email from the local host will be accepted.

By means of the /sbin/SuSEconfig script, both settings and the entries in the file /etc/sysconfig/postfix are translated into suitable parameters in the file /etc/postfix/main.cf.

If you do not want **SuSEconfig** to generate this configuration file, set the variable MAIL_CREATE_CONFIG in the file /etc/sysconfig/mail to **no**.

To configure Postfix, you need to know how to do the following:

- Configure Postfix Using /etc/sysconfig/postfix
- Configure Postfix with /etc/postfix/main.cf

Configure Postfix Using /etc/sysconfig/postfix

Modifications in the file /etc/sysconfig/postfix are only adopted in the file /etc/postfix/main.cf and, in some cases, in the file /etc/postfix/master.cf after the execution of /sbin/SuSEconfig or the SuSEconfig module for Postfix:

- **/sbin/conf.d/SuSEconfig.postfix**

  or

- **/sbin/SuSEconfig --module postfix**

The meanings of the variables are briefly commented on the configuration file /etc/sysconfig/postfix.

The following provides a more detailed description.

- **POSTFIX_RELAYHOST**. If the local email server should use a relay host to deliver emails that cannot be locally delivered, the relay host itself or the domain of the relay host must be given here.

If the name of a domain is provided, Postfix determines the relay host for the domain by an MX lookup.

If Postfix should forward all emails that cannot be locally delivered to a relay host without carrying out an MX lookup, the host name of the relay must be given in square brackets (for example, [**mailrelay.digitalairlines.com**]).

It is also possible to give an IP address in this form.

Optionally, the domain or host can be extended with a port number (for example, **digitalairlines.com:1025**).

If you leave this entry empty, Postfix delivers all mails that cannot be delivered locally to the mail exchanger.

Any entries in the file /etc/postfix/transport have precedence over the relay host.

If this variable is assigned a value, the variable relayhost in the file /etc/postfix/main.cf will be modified by running **SuSEconfig**.

- **POSTFIX_MASQUERADE_DOMAIN**. If your own DNS domain is configured with this variable (for example, digitalairlines.com), all addresses in emails that contain a host prefix are shortened by this host prefix.

  For example, geeko@da2.digitalairlines.com becomes geeko@digitalairlines.com.

  If this variable is assigned a value, the variable masquerade_domains in the file /etc/postfix/main.cf is modified by running **SuSEconfig**.

  Additionally, the variable masquerade_exceptions = root will be set.

- **POSTFIX_LOCALDOMAINS**. Contains a comma-separated list of the domains for which Postfix should accept emails.

  These values are written to the variable mydestination in the file /etc/postfix/main.cf by running **SuSEconfig**.

If POSTFIX_LOCALDOMAINS is empty, the variable is set to **$*myhostname*, localhost.$*mydomain*** by SUSEconfig.

- **POSTFIX_NULLCLIENT**. A nullclient is a host that can only send mail over the network, does not receive mail over the network, and does not deliver any mail locally.

  If you enter **yes**, the variable mydestination in the file /etc/postfix/main.cf will remain empty after running **SuSEconfig**.

  The default entry is **no**.

- **POSTFIX_DIALUP**. If this value is set to **yes**, emails that cannot be delivered locally are not sent to their destination until the command **sendmail -q** is run.

  The setting is useful for dial-up systems; otherwise, error messages would appear when sending emails if the system is not online, or a connection would be established for every email message if dial-on-demand is used.

  The value **no** leads to an immediate attempt to deliver any emails waiting for delivery.

  If this variable is assigned the value **yes**, the line defer_transports = smtp will be added to the file /etc/postfix/main.cf by running **SuSEconfig**.

- **POSTFIX_NODNS**: If this variable is set to **yes**, Postfix will not carry out any DNS lookups for the sender and recipient domains when processing emails.

  If this variable is assigned the value **yes**, the variable disable_dns_lookups = yes in the file /etc/postfix/main.cf will be activated by running **SuSEconfig**.

- **POSTFIX_CHROOT**. If this variable is set to **yes**, the services will be run in a chroot environment, if possible. You can find the chroot environment in /var/spool/postfix.

  If the variable is set to **no** (default), all Postfix processes will run in the normal environment.

- **POSTFIX_UPDATE_CHROOT_JAIL**. If SuSEconfig is to set up the chroot environment, this value should be set to **yes**.

  By default, the variable is set to **no**.

- **POSTFIX_LAPTOP**. Some Postfix services access FIFOs frequently, thus preventing the hard disk from spinning down.

  However, if this is desired on notebooks for power-saving purposes, the variable can be set to **yes**.

- **POSTFIX_UPDATE_MAPS**. If SuSEconfig is to create the database files from the corresponding lookup tables, this variable should be set to **yes** (default).

- **POSTFIX_MAP_LIST**. If POSTFIX_UPDATE_MAPS is set to yes, you can select the lists Postfix should support here.

- **POSTFIX_RBL_HOSTS**. Here you can specify a comma-separated list of host names from which RBLs (Realtime Blackhole List) can be obtained.

  No mail is accepted from clients that are these lists.

  This entry makes sense only if POSTFIX_BASIC_SPAM_PREVENTION is ***not*** set to **off**.

- **POSTFIX_BASIC_SPAM_PREVENTION**. Here, specify how strict filter rules for UCE (unsolicited commercial email) should be configured.

  Possible levels are **off**, **medium**, and **hard**.

  More details you can find at http://www.postfix.org/uce.html.

- **POSTFIX_MDA**. Here, specify an MDA with which Postfix should cooperate.

  The entries are

  - **procmail.** Use Procmail to deliver mail locally.
  - **cyrus**. Use lmtp to deliver to cyrus-imapd.
  - **local.** Use Postfix local MDA.

- **POSTFIX_SMTP_AUTH_***. These variables control the behavior of Postfix with respect to the authentication: if Postfix accepts mail and if Postfix delivers mail to other mail servers.
- **POSTFIX_SMTP_TLS_SERVER, POSTFIX_SMTP_TLS_CLIENT**. If these variables are set to **yes**, Postfix can encrypt the communication with the other side when sending and receiving mail, provided the following variables are configured.
- **POSTFIX_SSL_***, **POSTFIX_TLS_***. These variables control various aspects of the certificate and key management needed for the encryption.

  Encrypted connections are not covered in this course; this manual does not provide any details about the individual variables.
- **POSTFIX_ADD_***: These variables can be used to set the Postfix variables.

  The variable must be converted to uppercase letters and appended to POSTFIX_ADD_.

  For example, to set the Postfix variable message_size_limit to 100000, enter
  POSTFIX_ADD_MESSAGE_SIZE_LIMIT=100000

  in /etc/sysconfig/postfix.

  Subsequently, SuSEconfig will generate the respective entry message_size_limit=100000 in /etc/postfix/main.cf.

  All available Postfix variables can be listed by using **postconf**.
- **POSTFIX_REGISTER_SLP**. If this is set to yes, Postfix registers automatically to SLP.

Apart from this method, further settings can be made directly in the file /etc/postfix/main.cf, which has very detailed comments. Following a manual modification of the file /etc/postfix/main.cf, modifying /etc/sysconfig/postfix and subsequently running of **/sbin/SuSEconfig** will not affect the file /etc/postfix/main.cf.

Instead, the file /etc/postfix/main.cf.SuSEconfig will be created, which can be renamed to /etc/postfix/main.cf if necessary.

Configure Postfix with /etc/postfix/main.cf

The main configuration file for Postfix is

/etc/postfix/main.cf

This file is well documented, including detailed comments.

If you decide to configure Postfix directly by editing the configuration file /etc/postfix/main.cf, set the variable MAIL_CREATE_CONFIG in /etc/postfix/mail to **no**.

This will prevent SuSEconfig from overwriting the configuration file.

In case there are multiple lines containing settings for variables, the settings of the last definition will be used. This allows putting all your configuration lines at the end of the configuration file.

Some important variables are the following:

- **queue_directory**. The directory in which the mail queue is located. The default entry for this is /var/spool/postfix.
- **command_directory**. The directory in which the Postfix administration tools are located.

  The default entry is **/usr/sbin**.
- **daemon_directory**. The directory in which the Postfix daemon is located.

  The default entry is **/usr/lib/postfix**.
- **mail_owner**. Describes the owner of the mail queue.

  By default, this is set to **postfix**.

- **myhostname**. Defines the host name of the computer. This value serves later as the default value for other parameters.

  By default, the FQDN is given here.

- **mydomain**. The domain name of the computer.

  This value serves later as the default value for other parameters.

- **myorigin**. The domain that appears as the sender for emails sent locally.

  The default value is the FQDN.

- **mydestination**. Describes a list of domains for which the computer should accept emails.

- **masquerade_domains**. For sender addresses of the specified domain(s), the host part is removed.

  For example, geeko@da2.digitalairlines.com becomes geeko@digitalairlines.com.

- **masquerade_exeptions**. Specifies the users that should not be masqueraded. By default root is entered here.

- **relayhost**. All emails that cannot be processed locally are sent to the computer specified here.

- **inet_interfaces**. Specifies the network addresses on which Postfix waits for incoming mail.

  The default value is 127.0.0.1.

  To enable Postfix to receive mail from other hosts, enter the IP numbers of the network cards or **all**.

- **mynetworks.** Lists IP ranges belonging to your network.

  Postfix can be configured to forward mail from hosts in these networks.

  If you don't want to specify the IP ranges of your network by hand, you can use the option **mynetworks_style** which allows three values:

  - **class**. Postfix trusts all SMTP clients in the same IP class (A/B/C).
  - **subnet**. Postfix trusts all SMTP clients in the same IP subnet.
  - **host**. Postfix trusts only the local host.
- **smtpd_recipient_restrictions**, **smtpd_helo_restrictions**, **smtpd_client_restrictions**, **smtpd_sender_restrictions**. Control who is allowed to forward email over the mail server.

The variables that are relevant for most deployment scenarios are in the file

/etc/postfix/main.cf

Variables that are not defined here are assigned default values or remain empty.

To list all variables used by Postfix and their respective values, enter

postconf

Configure General Scenarios

The following scenarios presume that the variable MAIL_CREATE_CONFIG in the file /etc/sysconfig/mail is set to **no**.

If it is, the file /etc/postfix/main.cf will not be changed by executing **SuSEconfig**, and the file /etc/postfix/main.cf.SuSEconfig will not be generated.

Because these files usually contain useful settings, only few modifications are necessary for some deployment scenarios.

However, remember that the ***last*** entry of a variable in the file /etc/postfix/main.cf is valid.

If an entry is changed, the change does not take effect if a different value is assigned later in the file.

The following topics are described:

- Forward Mail to the Provider’s Mail Server
- Receive Mail over the Internet

Forward Mail to the Provider’s Mail Server

If all mail traffic is running from a mail server at the ISP, a small network merely needs a mail server that accepts the mail from the clients and passes it to the ISP’s mail server.

Because the local mail server does not serve as the mail server for the company domain from the Internet, the configuration is rather simple.

Such a mail server has to

- Accept mail from the intranet clients.
- Reject mail delivered by other clients.
- Possibly rewrite sender addresses.
- Submit all mail to the provider’s mail server.

Only few changes are needed in the file /etc/postfix/main.cf.

The following entries merely ensure that Postfix only accepts mail from the clients in the local network:

```
# 10.0.0.51 is the IP in the LAN
inet_interfaces = 10.0.0.51, 127.0.0.1
mynetworks = 10.0.0.0/24, 127.0.0.0/8
smtpd_recipient_restrictions = permit_mynetworks, reject
```

It is necessary to rewrite addresses to make sure that the sender does not appear in the form

geeko@da51.digitalairlines.com

but in the common form

geeko@digitalairlines.com

On the other hand, the host is important for messages sent to root.

Therefore, mail addressed to root should not be rewritten.

Two entries in the file /etc/postfix/main.cf are sufficient for this simple scenario:

```
masquerade_exceptions = root
masquerade_domains = digitalairlines.com
```

Moroever, Postfix must be informed of the mail server to which it is supposed to deliver the mail.

The relayhost entry also ensures that Postfix does not attempt to establish a direct contact to respective mail servers of the recipients.

```
relayhost = da1.digitailairlines.com
```

Exercise 4-2 Send Mail in the Local Network

In this exercise, you send mail in the local network. You configure Postfix and test your configuration.

You will find this exercise in the workbook.

(End of Exercise)

Receive Mail over the Internet

If the mail server is set up not only for sending email messages of the users in the local network but also for receiving mail from the Internet addressed to the domain, configuring it is a bit more difficult.

It is important to prevent the mail server from being misused as an open relay by spammers.

Regardless of the individual configuration of Postfix, the server must be introduced to the DNS as the responsible mail server by means of an MX record.

In addition to the requirements in the last section, the mail server has to

- Accept mail that comes from the Internet and is addressed to your domain
- Reject mail that comes from the Internet and is not addressed to your domain
- Reject mail from known spam sources

Accordingly, a number of additional entries are needed.

As mail can theoretically be received at all interfaces, a different value is necessary for inet_interfaces. mynetworks_style can remain unchanged:

```
inet_interfaces = all
mynetworks_style = subnet
```

Postfix has to know for which domains it is can accept mail:

```
myhostname = da51.digitalairlines.com
mydomain = digitalairlines.com
mydestination = $myhostname, localhost.$mydomain,
$mydomain
```

If Postfix is not only responsible for the mail of your domain but also for the mail of other domains (as is normally the case with web hosters), the domains are not entered under mydestination but in the lookup table **virtual**, which is covered in following section.

The decision to accept or not accept mail is controlled by the following variables, which contain various criteria.

- smtpd_helo_restrictions
- smtpd_sender_restrictions
- smtpd_recipient_restrictions
- smtpd_client_restrictions

A message is only delivered if it passes all the criteria without being rejected.

For example, smtpd_sender_restrictions can be used to prevent known spammers from delivering mail.

If the sender is listed in an RBL, the message can be rejected before the system checks whether it is addressed to a local user:

```
maps_rbl_domains = rbl-domains.digitalairlines.com
smtpd_sender_restrictions = reject_maps_rbl
```

The following entry ensures that email from the range specified in $mynetworks as well as email for which Postfix is responsible due to the specifications in $mydomain is accepted—all other mail is rejected due to reject_unauth_destination:

```
smtpd_recipient_restrictions = permit_mynetworks,
reject_unauth_destination
```

An explanation of all possibilities of the restrictions variables would exceed the scope of this course.

Exercise 4-3 Use Postfix on the Internet

In this exercise, you configure Postfix to send email to the Internet.

You will find this exercise in the workbook.

(End of Exercise)

Configure the Lookup Tables

Lookup tables contain rules for processing email within the overall Postfix system.

These tables are activated by variables in the file

/etc/postfix/main.cf

The tables are then defined as

/etc/postfix/*lookup-table*

After a lookup table has been defined, it needs to be converted to the required format (usually in the form of a hash table) using the command **postmap**.

This is done by entering:

```
postmap hash:/etc/postfix/sender_canonical
```

The structure of lookup tables is subject to the following general rules:

- Blank lines or lines that begin with a # are not interpreted as command lines.
- Lines that begin with a space are regarded as a continuation of the previous line.

It is also possible to use regular expressions.

Instead of domain names, you also can use IP addresses.

A man page exists for every lookup table: **man 5 *lookup-table***.

The following lookup tables are described:

- The access Lookup Table
- The canonical Lookup Table
- The recipient_canonical Lookup Table
- The sender_canonical Lookup Table
- The relocated Lookup Table
- The transport Lookup Table
- The virtual Lookup Table
- The aliases Lookup Table

The access Lookup Table

You can use the /etc/postfix/access lookup table to reject or allow email from defined senders.

The smtpd daemon evaluates this table when email arrives.

The following topics are described:

- Activate the Lookup Table
- The access Lookup Table Format

Activate the Lookup Table

This function is activated in the file /etc/postfix/main.cf:

```
smtpd_sender_restrictions = hash:/etc/postfix/access
```

The access Lookup Table Format

Each line defines a rule that is evaluated by smtpd when an email arrives.

The rules are processed from top to bottom and the matching of rules ends when the first match occurs.

Each line consists of the definition of an email address in the first column and a defined action in the second column.

Possible values for email address patterns are

- ***user@domain***. Defines a filter for the specified email address.
- ***domain.name***. Defines a filter for all email addresses of the specified DNS domain.
- ***user@***. Defines a filter for all email addresses with the same user part.

Possible values for actions are

- **4*xx Text*, 5*xx Text***. Rejects email with the specified numerical code (see RFC821) and the defined text message.
- **REJECT**. Rejects the email with a generic error message.
- **OK**. Accepts the email.
- **DISCARD *optional text***. Makes sure that the email is discarded without an error message to the sender.

  The ***optional text*** appears in the log file. If no text is specified, a generic message appears in the log.

Examples:

```
postmaster@digitalairlines.com  OK
spam@hahaha.net                 550 We're fighting against spam!
194.95.93.10                    REJECT
```


See the man pages (**man 5 access**) for other possible actions.

The canonical Lookup Table

You can use the /etc/postfix/canonical lookup table to rewrite sender and recipient addresses of incoming and outgoing emails.

Both the header and the envelope are rewritten.

The cleanup daemon reads this table when an email arrives.

The following is described:

- Activate the Lookup Table
- The canonical Lookup Table Format

Activate the Lookup Table

This function is activated in the file /etc/postfix/main.cf:

```
canonical_maps = hash:/etc/postfix/canonical
```

The canonical Lookup Table Format

Each line defines a rule that is evaluated by smtpd when an email arrives.

The rules are processed from top to bottom and the matching of rules ends when the first match occurs.

Each line consists of the definition of an email address in the first column and a defined action in the second column.

Possible values for email address patterns are

- ***user@domain***. Defines a filter for the specified email address.
- ***user***. Defines a filter for all email addresses with the same user part, provided the domain part of the email is listed in one of the variables $myorigin, $mydestination, $inet_interfaces, or $proxy_interfaces in the /etc/postfix/main.cf file.
- ***@domain***. Defines a filter for all email addresses of the specified domain.

Possible values for action are

- ***user@domain***. Rewrites the email address to the value defined here.

Examples:

```
training@digitalairlines.com   geeko@digitalairlines.com
@slc.digitalairlines.com       slc@digitalairlines.com
```

If you want to convert sender addresses and recipient addresses in a different way, use

- **recipient_canonical** to convert the recipient addresses
- **sender_canonical** to convert the sender addresses

The recipient_canonical Lookup Table

You can use the /etc/postfix/recipient_canonical lookup table to convert recipient addresses of incoming and outgoing emails.

The cleanup daemon evaluates this table when an email arrives before the generic lookup table /etc/postfix/canonical is evaluated.

The following topics are described:

- Activate the Lookup Table
- The recipient_canonical Lookup Table Format

Activate the Lookup Table

This function is activated in the file /etc/postfix/main.cf by the entry

```
recipient_canonical_maps = hash:/etc/postfix/recipient_canonical
```

The recipient_canonical Lookup Table Format

Each line defines a rule that is evaluated by smtpd when an email arrives.

The rules are processed from top to bottom and the matching of rules ends when the first match occurs.

Each line consists of the definition of an email address in the first column and a defined action in the second column.

Possible values for email address patterns are

- ***user@domain***. Defines a filter for the specified email address.
- ***user***. Defines a filter for all email addresses with the same user part, provided the domain part of the email is listed in one of the variables $myorigin, $mydestination, $inet_interfaces, or $proxy_interfaces} of the file /etc/postfix/main.cf.
- ***@domain***. Defines a filter for all email addresses of the specified domain.

Possible values for actions are

- ***user@domain***. Rewrites the email addresses to the value defined here.

Examples:

```
geeko@digitalairlines.com    training@digitalairlines.com
@slc.digitalairlines.com       slc@digitalairlines.com
```

The sender_canonical Lookup Table

You can use the /etc/postfix/sender_canonical lookup table to rewrite sender addresses of incoming and outgoing emails (for outgoing email: ***login*@*host*.*internal***.com to ***firstname.surname*@*mycompany***.com).

The cleanup daemon reads this table when an email arrives before the generic lookup table /etc/postfix/canonical is read.

The following topics are described:

- Activate the Lookup Table
- The sender_canonical Lookup Table Format

Activate the Lookup Table

This function is activated in the file /etc/postfix/main.cf by the entry

```
sender_canonical_maps = hash:/etc/postfix/sender_canonical
```

The sender_canonical Lookup Table Format

Each line defines a rule that is evaluated by smtpd when an email arrives.

The rules are processed from top to bottom and the matching of rules ends when the first match occurs.

Each line consists of the definition of an email address in the first column and a defined action in the second column.

Possible values for email address patterns are

- ***user@domain***. Defines a filter for the specified email address.

- ***user***. Defines a filter for all email addresses with the same user part, provided the domain part of the email is listed in one of the variables $myorigin, $mydestination, $inet_interfaces, or $proxy_interfaces of the file /etc/postfix/main.cf.
- ***@domain***. Defines a filter for all email addresses of the specified domain.

Possible values for actions are

- ***user@domain***. Rewrites the email address to the value defined here.

Examples:

```
training@digitalairlines.com    geeko@digitalairlines.com
@slc.digitalairlines.com        slc@digitalairlines.com
```

The relocated Lookup Table

You can use the /etc/postfix/relocated lookup table to return the corresponding bounced email, with a note of the new address of the desired addressee, to senders of emails to users that no longer exist on this system.

The following topics are described:

- Activate the Lookup Table
- The relocated Lookup Table Format

Activate the Lookup Table

This function is activated in the file /etc/postfix/main.cf by the entry

```
relocated_maps = hash:/etc/postfix/relocated
```

The relocated Lookup Table Format

Each line defines a rule that is evaluated by smtpd when an email arrives.

The rules are processed from top to bottom and the matching of rules ends when the first match occurs.

Each line consists of a key field in the first column, which refers to the email address of the former recipient or defines this by means of a regular expression and contact information in the second column, which may contain a new email address of the recipient or other contact information.

Possible values for the key field are

- ***user@domain***. Defines a filter for the specified email address.
- ***user***. Defines a filter for all email addresses with the same user part, provided the domain part of the email is listed in one of the variables $myorigin, $mydestination, $inet_interfaces, or $proxy_interfaces of the file /etc/postfix/main.cf.
- ***@domain***. Defines a filter for all email addresses of the specified domain.

Possible values for contact information include any information (such as email address or telephone number) that will help someone reach the email addressee. The information is used in "user has moved to ***new_location***" bounce messages.

Examples:

```
geeko@digitalairlines.com    geeko@novell.com
tux@digitalairlines.com   Please call 1-800-PIRATES
```

The notifications of the mail server are sent by email to the sender:

```
<geeko@digitalairlines.com>: host da51.digitalairlines.com[10.0.0.51]
said: 550
    <geeko@digitalairlines.com>: Recipient address rejected: User has
moved to
    geeko@novell.com (in reply to RCPT TO command)

<tux@digitalairlines.com>: host da51.digitalairlines.com[10.0.0.51] said:
550
    <tux@digitalairlines.com>: Recipient address rejected: User has moved
to
    Please call 1-800-PIRATES (in reply to RCPT TO command)
```

The transport Lookup Table

You can use the /etc/postfix/transport lookup table to define email routing for special email address ranges.

The following is described:

- Activate the Lookup Table
- The transport Lookup Table Format

Activate the Lookup Table

This function is activated in the file /etc/postfix/main.cf by the entry

```
transport_maps = hash:/etc/postfix/transport
```

The transport Lookup Table Format

Each line defines a rule that is evaluated via the qmgr or the trivial-rewrite daemon before an email is sent.

The rules are processed from top to bottom and the matching of rules ends when the first match occurs.

Each line consists of the definition of a domain pattern in the first column and a defined transport path in the second column.

Possible values for the domain pattern are

- ***user@domain***. Email to the specified user is forwarded over the defined transport route.
- ***domain***. All email to the specified domains are forwarded via the defined transport path.
- ***.domain***. All email with subdomains under the specified domain are forwarded via the defined transport path. This is only important if transport_maps is not listed in the variable parent_domain_matches_subdomain; otherwise, ***domain*** also includes ***.domain***.

Possible values for the transport path are

- ***transport*:*nexthop***. Different values can be assigned to ***transport***, such as **local**, **smtp**, or **uucp**. Also, any transport path can be assigned to ***transport***, including self-defined paths (such as Cyrus and Procmail).
  - **local**. Defines the delivery of email via the Postfix process local that delivers the email in the local system.

    For this specification, the value for ***:nexthop*** remains blank.
  - **smtp**. Defines the delivery of email via the Postfix process smtp, which delivers the email to a remote mail exchanger via the SMTP protocol.

    host or ***host:port*** can be configured as ***nexthop*** for an email exchanger on a remote host in case it does not accept email on port 25/TCP.

    To prevent DNS lookups on the MX entry, the form **[*host*]** or **[*host*]:*port*** should be used for the ***nexthop*** entry.

- **uucp**. Defines the delivery of email via the Postfix process pipe, which is configured by means of the file /etc/postfix/master.cf for the delivery of email via UUCP.

  The recipient host is specified as ***nexthop***.

Examples:

```
digitalairlines.com     smtp:da51.digitalairlines.com:10025
suse.com                 uucp:da150
```

The virtual Lookup Table

You can use the /etc/postfix/virtual lookup table to set up email for a number of domains with separate user names.

The following topics are described:

- Activate the Lookup Table
- The virtual Lookup Table Format

Activate the Lookup Table

This function is activated in the file /etc/postfix/main.cf by the entry

```
virtual_maps = hash:/etc/postfix/virtual
```

The virtual Lookup Table Format

Each line defines a rule that is evaluated via smtpd when an email arrives.

The rules are processed from top to bottom and the matching of rules ends when the first match occurs.

Using virtual domains requires the definition of the virtual domain first. This is done by placing the virtual domain name in the first column and arbitrary text in the second column. This text is only used to keep the structure of the file and has no meaning.

Every other line describing a recipient address of this domain contains

- First column: The recipient address.
- Second column:
  - The user name of the local email user to whom the incoming email should be delivered.

    or
  - A comma-separated list of all local email users to whom incoming emails should be delivered.

When you specify a virtual domain, only email addresses containing this virtual domain are modified. Address with a subdomain or host name are not modified. You need to specify them as virtual domains first.

Example:

```
virtual.domain                 geeko, tux
postmaster@virtual.domain      postmaster
user1@virtual.domain           geeko
user2@virtual.domain           tux
```


The aliases Lookup Table

The /etc/aliases lookup table is used to define aliases. You cannot redirect emails to mailboxes on other hosts or domains.

The following topics are described:

- Activate the Lookup Table
- The aliases Lookup Table Format

Activate the Lookup Table

This function is activated in the file /etc/postfix/main.cf by the entry

```
alias_maps = hash:/etc/aliases
```

The aliases Lookup Table Format

Each line defines a rule that is evaluated by smtpd when an email arrives.

The rules are processed from top to bottom and the matching of rules ends when the first match occurs.

Each line contains

- First column: A local recipient address followed by a colon.
- Second column: Filtered email is then redirected to another email user or to another email alias.

  Details of the target recipient in the second column can also be extended to include multiple recipients using a comma-separated list.

  An email is delivered explicitly to a local user if the recipient address in the second column begins with a "\".

The following is an example:.

```
root:           \root, geeko
mailer-daemon:  root
postmaster:     mailer-daemon
daemon:         root
webmaster:      tux@digitalairlines.com
wwwrun:         webmaster
```

If the file /etc/aliases has been modified, it must be converted into the hash table /etc/aliases.db by entering

```
da51:~ # postalias /etc/aliases
```

or

```
da51:~ # newaliases
```

Exercise 4-4 Use Lookup Tables

In this exercise, you use the Postfix lookup tables.

You will find this exercise in the workbook.

(End of Exercise)

Objective 6 Use Postfix Tools

Apart from the previously mentioned tools, Postfix also has a whole range of other useful administration tools that can make life considerably easier for a postmaster.

This section briefly introduces the administration tools for Postfix:

- **newaliases**. Converts the ASCII file /etc/aliases to the hash table /etc/aliases.db.
- **mailq**. Lists all emails in the mail queues that have not yet been sent.
- **postalias**. Converts the ASCII file /etc/aliases to the hash table /etc/aliases.db. Same as **newaliases**.
- **postcat.** Displays the contents of a file from the queue directories in a readable form.
- **postconf**. Without any parameters, this tool displays the values of all variables defined in the file /etc/postfix/main.cf as well as the values used by the standard variables. To modify variables directly, enter

  postconf -e *key=value*

  These changes are automatically integrated in the file main.cf.
- **postdrop**. This is run automatically by using the **sendmail** command, if **sendmail** cannot write any files to the maildrop directory because of missing world-writeable permissions. It saves the forwarded email as sgid maildrop.
- **postfix**. Enables configuration errors to be found (**postfix check**), forces email from the deferred queue to be delivered immediately (**postfix flush**), or rereads the Postfix configuration files (**postfix reload**).
- **postmap**. Generates the hash tables for the lookup tables in the directory /etc/postfix/.

- **postsuper**. Checks the file structure in the queue directories and removes unneeded files and directories (**postsuper -s**) or deletes files and directories that have been left after a system crash and are useless (**postsuper -p**).

  Individual email messages can be removed from the mail queues with **postsuper -d** ***ID***.

  In general, **postsuper** removes all files that are not normal files or directories (such as symbolic links).

Run the command **postsuper -s** immediately before starting the Postfix system.

For more information about these tools, see the man page **man 1** ***Postfix-Tool***.

Objective 7 Receive Email over IMAP and POP3

To retrieve email from the mail server, you can use two protocols:

- **IMAP** (Internet Message Access Protocol)
- **POP3** (Post Office Protocol)

Using IMAP, all emails are kept on the server. To read your email, you always have to establish a connection to the server. Using POP3, all email is transferred to the client. Email can be removed or kept on the server after they are transferred.

In this objective, you learn about the following:

- Configure Cyrus IMAPd
- Configure QPopper
- Sort Mail with Procmail

Configure Cyrus IMAPd

Cyrus IMAPd provides access to email via IMAP and POP3. The mailboxes of the users are stored in a database which uses a special format.

All users which receive email via this service need to exist on the computer. These users do not need to have full access to the system; it is sufficient to have a username and a password.

All email is stored in a Cyrus hierarchical file structure. Hierarchical means that changes on one level also influence the lower levels. It is possible for many servers to access this file structure at the same time.

Cyrus servers can handle many postboxes. Cyrus does not support relational databases like MySQL, but Cyrus does support quotas and ACLs.

To use Cyrus IMAPd, you need to know about the following:

- Install Cyrus IMAPd
- Configure Postfix to Use Cyrus IMAPd
- Cyrus IMAPd in Detail

Install Cyrus IMAPd

To install Cyrus IMAPd, select the package **cyrus-imapd**.

Configure Postfix to Use Cyrus IMAPd

You have to configure Postfix to use Cyrus IMAPd. The file /etc/postfix/master.cf already contains the following entry

```
cyrus     unix  -       n       n       -       -       pipe
  user=cyrus argv=/usr/lib/cyrus/bin/deliver -e -r ${sender} -m
${extension} ${user}
```

This calls the program **deliver**. This program submits all email to the recipients on the system. It replaces the program **local** of Postfix.

You only need to assign Cyrus the responsibility for a mail domain. You can do this, for example, in the file /etc/postfix/transport by appending the following line:

```
digitalairlines.com      cyrus:
```

The colon at the end of the line is required.

With this transport configuration also the emails to user root are forwarded to Cyrus. To prevent this add another line that /etc/postfix/transport looks like this:

```
root@                    local:
digitalairlines.com      cyrus:
```

To create the corresponding lookup table, use

postmap hash:/etc/postfix/transport

Now, Postfix will give all email for the domain digitalairlines.com to the program **deliver** of the Cyrus package.

Cyrus IMAPd in Detail

The most important files and directories of Cyrus IMAPd are described in the following:

- **/etc/cyrus.conf**. The basic configuration file of Cyrus IMAPd. Normally it is not necessary to make changes.
- **/etc/imapd.conf**. The configuration file of the IMAPd.
- **/var/lib/imap/**. This directory contains the variable files of Cyrus IMAPd. This includes log files, database files, and information about the mailboxes of individual users.
- **/var/lib/imap/mailboxes.db**. This file includes the configuration of the mailboxes. It is important to back up this file at regular intervals using the cron service via the file /etc/cron.daily/cyrus.
- **/var/lib/imap/log/**. This log stores information. The file name of the log file is equivalent to the process number of the IMAP server. If you use the syslog daemon for logging, you do not need the files in this directory.
- **/var/lib/imap/quota/**. The quotas for each IMAP account can be added here.

- **/var/lib/imap/shutdown** If this file exists, the IMAP server sends the first line of this file as an alert to the client. The connection will be closed and no other clients can connect until this file is removed. This is useful for maintenance.
- **/var/lib/imap/msg/motd**. If this file exists, the IMAP server sends the first line to the client after the connection is created. This can be used for welcome messages.
- **/usr/lib/cyrus/**. The home directory of the user cyrus. The subdirectory bin/ contains most of the programs of the Cyrus IMAPd.

This objective discusses the following:

- The file /etc/imapd.conf
- Test the IMAPd
- Add Users to the System
- Administer Cyrus IMAPd

The file /etc/imapd.conf

The Cyrus server is configured in the file

/etc/imapd.conf

Each line matches the format

option*: *value

After Cyrus server installation, the file looks like the following:

```
configdirectory: /var/lib/imap
partition-default: /var/spool/imap
sievedir: /var/lib/sieve
admins: cyrus
allowanonymouslogin: no
autocreatequota: 10000
reject8bit: no
quotawarn: 90
timeout: 30
poptimeout: 10
dracinterval: 0
drachost: localhost
sasl_pwcheck_method: saslauthd
lmtp_overquota_perm_failure: no
lmtp_downcase_rcpt: yes
#
# if you want TLS, you have to generate certificates and keys
#
#tls_cert_file: /usr/ssl/certs/cert.pem
#tls_key_file: /usr/ssl/certs/skey.pem
#tls_ca_file: /usr/ssl/CA/CAcert.pem
#tls_ca_path: /usr/ssl/CA
```

The most important options are

- **configdirectory**. Location of the directory with the account configurations.
- **partition-default**. Location where the mailboxes are stored.
- **admins**. List of users that have administrative permissions. (Use a space to separate user names.) By default, the user cyrus is used for administration of the IMAP server.
- **allowanonymouslogin**. If this option is **yes**, you can log in without entering a user name and a password. This should never happen!
- **autocreatequota**. If this option is set to **0** (zero), the user is not allowed to create new mail folders.

  If this option is set to **-1**, there is no quota for users.

If the option is set to **1** or more, this value (in KB) is the default quota. The default value is 10000, allowing for 10 MB of email data to be stored in the mailbox of each user.

- **quotawarn**. Percentage of used quota space. When this percentage is reached, the server will warn the user that his space is running out.
- **timeout**. Inactivity time in minutes when the server will disconnect the client.
- **sasl_pwcheckmethod**. SASL (Simple Authentication and Security) allows authentication for connection-oriented protocols. This is more secure than using UNIX passwords. We recommend that you use **saslauthd**.

Test the IMAPd

Because Cyrus IMAPd uses SASL for authentication, this service has to be started first:
rcsaslauthd start

To start the Cyrus IMAPd, enter
rccyrus start

In order to have both services started automatically, use
insserv saslauthd
and
insserv cyrus

After starting the IMAPd, you can test the server by using **telnet**.

To close the telnet dialog, enter
. logout

The following is an example:

```
da51:~ # telnet localhost 143
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
* OK da51 Cyrus IMAP4 v2.2.12 server ready
. logout
* BYE LOGOUT received
. OK Completed
Connection closed by foreign host.
da51:~ #
```

Add Users to the System

To define mail boxes for users on the mail server, these users have to be created on the system. Because the mail boxes for these users are stored in the directory /var/lib/imap/, it is not necessary to provide the users with a home directory.

Only usernames and passwords have to exist in the files /etc/passwd and /etc/shadow. This information is needed when a user is receiving email from a client.

For security reasons, the users should not be allowed to log in to the mail server. Therefore, no home directory should be created and the login shell should be set to /usr/bin/passwd. When users log in to the mail server, they can only modify their passwords but cannot perform any other tasks on the system.

Adding a new user to the system is done with
useradd -s /usr/bin/passwd *username*
passwd *username*

Administer Cyrus IMAPd

In the /etc/imapd.conf file, the user cyrus is configured for administering the Cyrus server (option: **admins**).

This user is added to the system during Cyrus installation:

```
da51:~ # grep cyrus /etc/passwd
cyrus:x:96:12:User for
cyrus-imapd:/usr/lib/cyrus:/bin/bash
da51:~ #
```

This user accout does not have a system password yet, so you have to set it:

```
da51:~ # passwd cyrus
Changing password for cyrus.
New password:
Re-enter new password:
Password changed
```

Cyrus IMAPd is using SASL (Simple Authentication and Security Layer) for authentication. saslauthd is a daemon process that handles plaintext authentication requests on behalf of the SASL library.

The SASL server fulfills two roles:

- It isolates all code requiring superuser privileges into a single process.
- It can be used to provide proxy authentication services to clients that do not understand SASL-based authentication.

To administer your Cyrus server, you need the command **cyradm** to do all the other administration tasks. This tool is interactive.

To start **cyradm**, enter

```
da51:~ # cyradm -user cyrus -auth login localhost
```

You are prompted to enter the cyrus password. After this, the prompt changes to

```
localhost>
```

To see an overview of the possible cyradmin commands, enter

- **help**

  or simply

- **?**

The most important commands of **cyradm** are the following:

- **listmailbox** or **lm**. Lists the names of all mailboxes.
- **createmailbox** or **create** or **cm**. Creates a mailbox. The user must have a Linux account on the server.
- **deletemailbox** or **delete** or **dm**. Deletes a mailbox.
- **renamemailbox** or ***rename*** or **renm**. Renames or moves a mailbox.
- **setquota** or **sq.** Sets a quota for a mailbox or resource.
- **listquota** or **lq.** Lists the quota for a mailbox or resource.
- **listquotaroot** or **lqr** or **lqm**. Lists the quota for the super folder or resource.
- **setaclmailbox** or **setacl** or **sam**. Adds a new ACL to a mailbox.
- **deleteaclmailbox** or **deleteacl** or **dam**. Deletes an existing ACL.
- **listaclmailbox** or **listacl** or **lam**. Lists all ACLs of a mailbox.

The following is an example:

```
localhost> createmailbox user.geeko
localhost> listmailbox
user.geeko (\HasNoChildren)
localhost> deletemailbox user.geeko
deletemailbox: Permission denied
localhost> setaclmailbox user.geeko cyrus c
localhost> deletemailbox user.geeko
localhost>
```

In the first line, a new mailbox for the user geeko is created.

The names of the mailboxes have to start with “user.” in order to have the correct permissions set.

---

listmailbox shows the new mailbox.

Deleting the new mailbox by using **deletemailbox** failed because the permission to delete a mailbox is not set. By default, no administrator has the permission to delete mailboxes. This is to prevent errors.

The permission to delete the mailbox is set by the **setaclmailbox** command.

To set quotas, use the quota commands listed above:

```
localhost> listquotaroot geeko

localhost> setquota geeko 200000
quota:200000
localhost> listquotaroot geeko
geeko STORAGE 0/200000 (0%)
```

All incoming email is stored in the mailboxes located in the directory /var/spool/imap/user/. Additional information (e.g., about reading email) is stored in subdirectories of /var/lib/imap/user/.

The following table describes ACL:

Table 4-5

ACL Flags	Description
l	Lookup (get the name of a mailbox)
r	Read (read the content of a mailbox)
s	Mark mail as seen
w	Write flags (other than seen and delete)
i	Insert (insert, copy, or move mails)
p	Post (send mail to mailbox)
c	Create and delete mailbox and sub-mailboxes
d	Mark as delete
a	Administer (set ACLs)

To simplify the use of these permissions, other abbreviations are available:

Table 4-6

Abbreviation	Permissions
none	None
read	lrs
post	lrps
append	lrsip
write	lrswipcd
all	lrswipcda

Each user can administrate his or her own mailbox using the **cyradm** command:

```
geeko@da51:~> cyradm -user geeko -auth login da51
IMAP Password:
localhost> createmailbox user.geeko.team
localhost> listmailbox
INBOX (\HasChildren)         INBOX.team (\HasNoChildren)
```

If user geeko wants to give user jgoldman permission to read her mailbox, geeko.team, she enters

```
localhost> listaclmailbox user.geeko.team
geeko lrswipcda
localhost> setaclmailbox user.geeko.team jgoldman read
localhost> listaclmailbox user.geeko.team
jgoldman lrs
geeko lrswipcda
```

All the settings and definitions are effective for receiving email via IMAP and POP3. The only difference is the setup of your MUA.

Exercise 4-5 Configure Cyrus IMAPd

In this exercise, you install and set up CyrusIMAPd.

You will find this exercise in the workbook.

(End of Exercise)

Configure QPopper

Because the configuration of Cyrus IMAPd is quite complex (you have to create mailboxes for every user in the IMAP database), you may choose to only offer POP3 as the protocol to provide email for your users. Using qpopper for this service is much easier than using Cyurs IMAPd for this.

POP (Post Office Protocol Version) is a protocol that allows an email client to poll (download) its email from a POP server. The current version of POP is POP3. POP3 has been expanded several times since its original definition in RFC1081 and is currently defined at great length in RFC1939.

POP3 was developed because not every host that wanted to both send and receive mails had enough resources (RAM, disk space, and CPU) to support a full mail server so it could directly receive its own email.

Many email providers do not provide their users with login access to the email server, instead they offer a POP3 server from which to get mail.

POP3 has the following characteristics:

- POP3 controls the transfer of mail, initiated by the recipient (as opposed to SMTP, where the sender initiates the transfer).
- The client makes a connection on server port 110 using TCP.
- The POP3 client and POP3 server exchange commands (client) and answers (server) until the connection is ended or aborted.
- All commands are given in plain text, including arguments and parameters where relevant. The server sends the client a status message after each command. This has either the value +OK (command successful) or -ERR (command could not successfully be run) and an optional text message.
- POP3 supports the authentication of the client by the server.

- After a successful authentication process, the client and server enter the transaction phase, in which one or more email messages can be transferred.
- When the connection is closed by the client, the server enters an update phase.

  The email messages are deleted from the mailbox and resources are released.

This objective explains how to do the following:

- Install QPopper
- Configure QPopper

Install QPopper

A POP3 server (QPopper) is supplied with SUSE Linux Enterprise Server 10. If SUSE Linux Enterprise Server 10 is not yet installed, use YaST to select and install the qpopper package.

Configure QPopper

As soon as QPopper is installed, you can use it to retrieve mail with the POP3 protocol, if the superdaemon xinetd has been configured and started in the system.

The POP3 server is started by xinetd as soon as it receives a TCP request on port 110. The configuration of xinetd is described in the Novell Course ***SUSE Linux Administration*** (3072).

The entry in the file /etc/xinetd.d/qpopper needs to look like the following:

```
#
# qpopper - pop3 mail daemon
#
service pop3
{
#       disable         = yes
        socket_type     = stream
        protocol        = tcp
        wait            = no
        user            = root
        server          = /usr/sbin/popper
        server_args     = -s
        flags           = IPv4
}
```

The first line in the service environment (**disable = yes**) is commented out. Alternatively you can set this parameter to **no**.

If xinetd is not running, it can be started on a running system by entering

rcxinetd start

To start xinetd automatically each time the system is booted, enter

insserv xinetd

If qpopper is activated, it provides the email from the directory /var/spool/mail/ via POP3.

With this configuration, the POP3 server is ready for operation and will process all requests from POP3 clients.

For more information on QPopper see man page (**man 8 popper**) in the /usr/share/doc/packages/qpopper/ directory.

It is normally not a good idea for users to be able to log in to the computer functioning as the POP3 server and to read their email there, but they must be able to change their passwords there at all times. To implement this, the login shell for the user in the file /etc/passwd is set to /usr/bin/passwd:

geeko:x:1000:100:Geeko Chameleon:/home/geeko:/usr/bin/passwd

This can be set by entering the command

usermod -s /usr/bin/passwd geeko

Exercise 4-6 Configure QPopper

In this exercise, you install and activate QPopper.

You will find this exercise in the workbook.

(End of Exercise)

Sort Mail with Procmail

Procmail is currently the most widely used MDA (Mail Delivery Agent) under Linux. Procmail is installed in a standard SUSE Linux Enterprise Server 10 installation. Postfix (and also Sendmail) integrate Procmail as an MDA.

Procmail is a highly flexible MDA.

It can sort email into mailboxes, forward them to other recipients, or delet them according to almost any arbitrary criteria a user specifies.

In particular, Procmail is extremely good at sorting large quantities of mail and automatically disposing of unwanted email.

This objective describes how to do the following:

- Install Procmail
- Configure Procmail
- Filter and Distribute Incoming Email

Install Procmail

Procmail (package **procmail**) is automatically installed during the installation of Postfix.

Configure Procmail

If Postfix is used as the MTA, the following entry is required in /etc/postfix/main.cf:

```
mailbox_command = /usr/bin/procmail
```

Then, Postfix passes received email to Procmail for local delivery.

It is not necessary to take any further configuration steps to get Procmail to deliver email to the mailbox of the respective user.

For more information about all possible Procmail options, see man pages **man 1 procmail** and **man 1 procmailrc**.

Filter and Distribute Incoming Email

The most interesting aspect of using Procmail is the prefiltering and distribution of email based on the information in the mail header or the email body.

This is Procmail's great strength—it offers more filtering options than any other MDA.

You can set up separate filters for each mail recipient.

Each email recipient can create a file .procmailrc in his or her home directory and install his or her own desired settings for Procmail. It is possible also to create a system-wide configuration, using the configuration file /etc/procmailrc.

Where these configuration files do not yet exist or contain no commands for Procmail, all mail is delivered to the /var/spool/mail/***username*** mailbox.

If incoming email should be distributed by Procmail to several different mailboxes, it must ensure that the directories containing the mailboxes exist.

Example 1:

A typical configuration file for Procmail might have the following contents:

```
MAILDIR=$HOME/Mail
LOGFILE=$MAILDIR/mail.log

:0
* ^From.*digitalairlines
$MAILDIR/Digitalairlines

:0
*
$MAILDIR/Inbox
```

This configuration file causes Procmail to respond to incoming email as follows:

- The $MAILDIR environment variable is set to **$HOME/Mail**.

  Procmail interprets any further entries with a relative path in the configuration file relative to this directory.

- The $LOGFILE environment variable is set to **$MAILDIR/mail.log**.

  This is where log information is placed.

- The next three lines build a rule comprised of a key, a filter, and an action.
  - The line containing **:0** (the key) specifies that data from the email header will be compared with the filter expression in the following line.

    When a match is found, the email will be processed in accordance with the last line.

    Procmail will not process the mail further, even when it matches further rules.

- The line *** ^From.*digitalairlines** compares all lines in the email header with the given regular expression **^From.*digitalairlines**.

  The comparison here is not case-sensitive. All email messages that contains **digitalairlines** in the From line match this filter.

- The line containing **$MAILDIR/Digitalairlines** instructs Procmail to place the corresponding email in this mail folder.

- The last three lines build another rule just like the first one.
  - All email that passes through this last rule is checked by the filter expression, *****.
  - Since no filter is given here, the rule matches all remaining email, and these are placed in the corresponding mail folder, in accordance with the last line (**$MAILDIR/Inbox**).

Example 2:

A more detailed configuration is shown in the following example:

```
MAILDIR=$HOME/Mail
LOGFILE=$MAILDIR/mail.log

:0
* ^From.*digitalairlines
* ^Subject.*mail
{
  :0 c
  $MAILDIR/Digitalairlines

  :0
  ! jgoldman@digitalairlines.com
}

:0
*
$MAILDIR/Inbox
```

This configuration file causes Procmail to process email as follows:

- The environment variables are set again in the first lines (see Example 1).
- Any email that contains **digitalairlines** in the From header line and contains **mail** in the subject line of the header (case-insensitive) is placed in the **$MAILDIR/Digitalairlines** mail folder.

  Procmail makes a copy of this email (**:0 c**) and sends it to **jgoldman@digitalairlines.com**.
- All email that doesn't match the above rule are matched against ***** (which matches all the remaining email) and these are sent to **$MAILDIR/Inbox**.

For more information on the ~/.procmailrc configuration file, use the manual page (**man 5 procmailrc**).
There is a good collection of useful configuration examples in the manual page of procmailex (**man 5 procmailex**).

In the directory /usr/share/doc/packages/procmail/examples/ some examples of .procmailrc files.

Exercise 4-7 Configure Procmail

In this exercise, you configure Procmail.

You will find this exercise in the workbook.

(End of Exercise)

Objective 8 Manage Spam

The most common tool used to filter spam in Linux is **SpamAssassin**. It tests the email body, header, URI and the attachments of mails.

It also uses black lists and white lists of other organizations (e.g., http://razor.sf.net, http://pyzor.sf.net).

Every hit is weighted with a number. If the sum reaches a defined value, the mail is marked as spam.

This objective explains the following:

- Install SpamAssassin
- Configure SpamAssassin
- Use SpamAssassin
- Test SpamAssassin
- Train SpamAssassin

Install SpamAssassin

Select the package **spamassassin** to install the spam filter.

After the installation, four executable files will be available:

- **spamassassin**. Starts SpamAssassin.
- **spamd**. A daemonized version of the SpamAssassin executable. This daemon is useful for automated mail checks.
- **spamc**. A client for spamd. Use spamc instead of SpamAssassin in your own scripts.
- **sa-learn**. A tool to train SpamAssassin.

Configure SpamAssassin

SpamAssassin is configured by editing the file

/etc/mail/spamassassin/local.cf

After the installation, this file is similar to the following:

```
# Add your own customisations to this file.  See 'man
Mail::SpamAssassin::Conf'
# for details of what can be tweaked.
#


# do not change the subject
# to change the subject, e.g. use
# rewrite_header Subject ****SPAM(_SCORE_)****
rewrite_header Subject

# Set the score required before a mail is considered spam.
# required_score 5.00

# uncomment, if you do not want spamassassin to create a new message
# in case of detecting spam
# report_safe 0
```

Some possible options are described:

- **required_score *number***. Threshold when an email is marked as spam (5.0=low, 7.5=medium, 10.0=high).
- **rewrite_header [subject|from|to] *string***. This ***string*** is added to the specified header entry.
- **report_save [0|1|2]**. If set to **1**, the spam mail is sent as an encapsulated attachment. If set to **2**, only text spam mails are sent as attachment.
- **use_bayes [0|1]**. If set to **1**, the statistical Bayes filter system is used.
- **bayes_auto_learn [0|1]**. If set to **1**, the Bayes filter learns what is spam and what is not.

- **skip_rbl_checks [0|1]**. If set to **1**, SpamAssassin uses RBLs (DNS black lists).
- **ok_locales** ***string***. Lists all the character sets SpamAssassin can receive (Chinese, English, Japanese, Korean, Russian, and Thai).

  If you want to allow all character sets, enter **all**.
- **whitelist_from @*domain***. Lists the domains that mail can be received from.

For more information on all possible options, see
man 3 Mail::SpamAssassin::Conf

The directory /usr/share/spamassassin/ contains a lot of filter rules for detecting the spam mails:

```
da51:~ # ls /usr/share/spamassassin/
10_misc.cf              20_uri_tests.cf      30_text_de.cf
20_advance_fee.cf       23_bayes.cf          30_text_fr.cf
20_anti_ratware.cf      25_accessdb.cf       30_text_it.cf
20_body_tests.cf        25_antivirus.cf      30_text_nl.cf
20_compensate.cf        25_body_tests_es.cf  30_text_pl.cf
20_dnsbl_tests.cf       25_body_tests_pl.cf  30_text_pt_br.cf
20_drugs.cf             25_dcc.cf            50_scores.cf
20_fake_helo_tests.cf   25_domainkeys.cf     60_awl.cf
20_head_tests.cf        25_hashcash.cf       60_whitelist.cf
20_html_tests.cf        25_pyzor.cf          60_whitelist_spf.cf
20_meta_tests.cf       25_razor2.cf         60_whitelist_subject.cf
20_net_tests.cf         25_replace.cf        languages
20_phrases.cf           25_spf.cf            sa-update-pubkey.txt
20_porn.cf              25_textcat.cf        triplets.txt
20_ratware.cf           25_uribl.cf          user_prefs.template
```


Use SpamAssassin

spamassassin (and **spamc**) expect their input from STDIN.

The simplest way to use SpamAssassin is to pipe the mailbox into the command **spamassassin**:

```
da51:~ # cat /var/spool/mail/root | spamassassin
```

If you want to use SpamAssassin with your Postfix configuration, the easiest way is to use Procmail. You need to edit only three files:

- Create the file /etc/procmailrc with this content:

```
#LOGFILE=/tmp/procmail.log
#VERBOSE=yes

# Until now, mail is untagged, you may add rules for
# mail that must not be tagged

:0 hbfw
| /usr/bin/spamc
```

  The mail is piped in the first filter rule to the spamc. The flags of the filter rule are explained below:

  - **h.** Feed the header to the pipe, file, or mail destination (default).
  - **b.** Feed the body to the pipe, file, or mail destination (default).´
  - **f.** Consider the pipe as a filter.
  - **w.** Wait for the filter or program to finish and check its exitcode (normally ignored); if the filter is unsuccessful, then the text will not have been filtered.

- Make sure that you activate Procmail in /etc/postfix/main.cf:

```
mailbox_command = /usr/bin/procmail
```

- Make sure that your smtpd definition in the file /etc/postfix/master.cf is set to default

```
smtp      inet  n       -       n       -       -       smtpd
```

After this, start spamd by entering

rcspamd start

Spam can be filtered from the mail client or in a further Procmail configuration (~/.procmailrc) by adding lines similar to this:

```
:0
* ^X-Spam-Status: Yes
$MAILDIR/Spam
```

Test SpamAssassin

You can use **telnet** to test your configuration. You also can send a spam email directly using the **sendmail** command:

```
cat sample-spam.txt | /usr/sbin/sendmail geeko@digitalairlins.com
```

Train SpamAssassin

SpamAssassin can learn and improve its spam detection quality. Therefore, it is useful to collect all the undetected spam in a separate folder.

To teach SpamAssassin what spam is, enter

sa-learn --spam --file *mailfolder*

To teach SpamAssassin what is not spam, enter

sa-learn --ham --file *mailfolder*

For more information on sa-learn, see the man page **man 1 sa-learn**.

Exercise 4-8 Manage Spam with SpamAssassin

In this exercise, you install and configure SpamAssassin to manage spam.

You will find this exercise in the workbook.

(End of Exercise)

Objective 9 Use a Virus Scanner for Email

Communication via email is very important for companies and individuals today, but email can be infected by virus software.

Most of the viruses attack Windows clients, but the mail server of a company is often a Linux or UNIX machine. To avoid damage we recommend to search for viruses before the infected mail arrives at the user's client.

SUSE Linux Enterprise Server 10 provides tools to detect viruses in email on your mail server.

In this objective, the following tools are introduced:

- AVMailGate
- AMaViSd-new

AVMailGate

AVMailGate is the abbreviation for AntiVir MailGate, an antivirus mail filter from H+BEDV Datentechnik GmbH (http://www.hbedv.com).

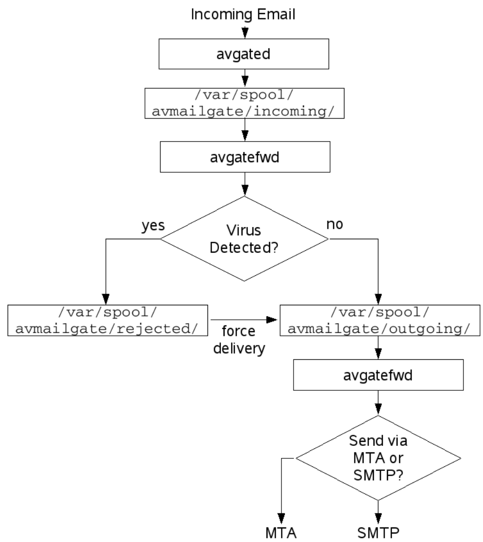
AVMailGate updates the virus definition file and the engine itself.

This topic describes the following:

- AVMailGate Architecture
- Install AVMailGate
- Configure AVMailGate
- Understand the AVMailGate Configuration Files
- Update AVMailGate

AVMailGate Architecture

Figure 4-5



AVMailGate is composed of three queues and two kinds of processes:

- **Queues** (located in /var/spool/avmailgate/)
    - **incoming/.** The input queue for all incoming email.
    - **rejected/.** The queue where possible infected email are stored.
    - **outgoing/.** The output queue for not infected email.
- **Processes**
    - **avgated.** The smtpd receiver that stores incoming email in the input queue. (daemon)
    - **avgatefwd.** Virus-scanning function and SMTP forwarder and sendmail invoker. (daemon)

Both processes can be configured by editing the file **/etc/avmailgate.conf**

This file includes comments that describe the possible settings.

For a description of all options, see /usr/share/packages/avmailgate/doc/MANUAL.

Install AVMailGate

To install AVMailGate, select the package **avmailgate**.

A startscipt is available in **rcavgate** to start and stop AVMailGate.

To start AVMailGate at the system's start, enter **insserv avgate**.

Configure AVMailGate

There are two possibilities to use AVMailGate with Postfix:

- AVMailGate Waits for Mails on Port 25 and Forwards Them to Postfix
- AVMailGate Is Used from Postfix as Content Filter

AVMailGate Waits for Mails on Port 25 and Forwards Them to Postfix

To use AVMailGate as mail proxy that waits for mails on port 25 you simply have to avoid that Postfix listens on this port. Therefore mark the following line in /etc/postfix/master.cf: as command by adding a hash ("#") in front

```
#smtp      inet  n       -       n       -       -       smtpd
```

Then restart Postfix.

AVMailGate Is Used from Postfix as Content Filter

The best way to use AVMailGate with Postfix is to use the Full Content Filter API (see FILTER_README in the Postfix package for details).

It waits for SMTP connections on port 10024 and sends scanned email messages or virus warnings out per SMTP on port 10025.

To integrate AVMailGate as a filter in Postfix, do the following:

1. Edit /etc/postfix/master.cf and uncomment –if applicable– or add the following line:

```
localhost:10025 inet n  -     y     -     -    smtpd -o
content_filter=
```

2. Edit /etc/postfix/main.cf and add the following line:

```
content_filter = smtp:127.0.0.1:10024
```

3. Since AVMailGate listens on port 10024, not port 25, edit /etc/avmailgate.conf and set

```
ListenAddress    localhost    port 10024
```

4. You have to tell AVMailGate it should send email back to Postfix via SMTP on host localhost via port 10025.

   Edit /etc/avmailgate.conf and set

```
ForwardTo    SMTP: localhost port 10025
```

5. Since AVMailGate sends out notification messages as AVMailGate, set an alias in /etc/aliases:

```
vmailgate:    root
```

You must run **newaliases** afterward.

6. After these changes, enter

   rcpostfix reload

   and

   rcavgate start

Your system is now ready to scan email.

Understand the AVMailGate Configuration Files

As mentioned before, the configuration file of AVMailGate is /etc/avmailgate.conf. The options of this file can be grouped:

- General Parameters
- Scanning of Files in an Archive
- Handling Envelope Recipient Addresses
- Adding a Notification in the Body of Transmitted Mails
- Other Configuration Files

General Parameters

- **User** and **Group**. avgated and avgatefwd run with the permissions of this user and group.
- **Postmaster**. Errors and alert messages are sent to this user.
- **MyHostName**. Hostname of the computer. If not set, it is retrieved by gethostname.
- **SpoolDir**. The directory where the queues are stored.
- **AntiVirDir**. Directory where the virus signatures are stored.
- **TemporaryDir**. Temporary directory where email messages will be extracted and scanned.

- **PidDir**. The location of the PID files.
- **LogFile**. Location of the log file.
- **SyslogFacility**. Facility argument for the syslog daemon (default: **mail**).
- **MaxIncomingConnections**. Maximum number of simultaneous connections.
- **SMTPTimeout**. Number of seconds until an SMTP timeout occurs.

  More detailed timeouts can be configured by using the following options:
  - **SMTPGreetingTimeout**
  - **SMTPHeloTimeout**
  - **SMTPMailFromTimeout**
  - **SMTPRcptTimeout**
  - **SMTPDataTimeout**
  - **SMTPDataBlockTimeout**
  - **SMTPDataPeriodTimeout**
- **MaxMessageSize**. Maximum size of a message in bytes.
- **MaxRecipientsPerMessage**. Maximum number of recipients.
- **MinFreeBlocks**. Number of free file system blocks. If the limit is reached, no more incomming email is accepted. (0=disable feature)
- **MaxForwarders**. Maximum number of forward processes of avgatefwd. The number depends on the quality of the network connection (low network quality > higher value).
- **BlockSuspiciousMime**. If set to **YES**, suspicious MIME email will be blocked.
- **BlockExtensions**. Filename extensions that should be blocked (separated by a semicolon).

- **ExposeRecipientsAlerts**, **ExposeSenderAlerts**, **ExposePostmasterAlerts**. Specifies if alerts will be sent to the recipient, sender, or postmaster. The possible values are
  - **NO**. No alerts will be sent.
  - **LOCAL** (not available for **ExposePostmasterAlerts**). Alerts will be sent if the recipient/sender is a local user.
  - **YES**. Alerts will be sent.
- **AlertsUser**. Name of sender of alerts. (Syntax: **username** or **username@domain**)
- **ListenAddress**. Interface and port the SMTP daemon listens on. The default interface 0.0.0.0 means all interfaces.

  Syntax: **ListenAddress *interface* port *port***
- **ForwardTo**. Type of mail forwarding.
  1. By piping: **ForwardTo *sendmail***
  2. By SMTP: **ForwardTo SMTP: *host* port *port***

  Default: **ForwardTo /usr/sbin/sendmail -oem -oi**
- **RefuseEmptyMailFrom**. If set to **YES**, mails containing a blank sender address will be blocked.
- **PollPeriod**. Interval (in seconds) of queue scanning done by avgatefwd.
- **QueueLifetime**. Maximum time a message can stay in the queue before it will be bounced. (0 = disable feature)
- **ForwarderRetryDelay**. Maximum time between retrying to send a queued message.
- **ThrottleMessageCount**. Number of messages that will be reprocessed in a given time. (Only needed for large queues.)

  After reprocessing, the avgatefwd will sleep for **ThrottleDelay** seconds.

- **BounceMessageUser**. Name or mail address of the sender of error messages (e.g., if an email could not be delivered).

Scanning of Files in an Archive

- **ArchiveMaxRecursion**. Maximum of recursion depth of unpacking and scanning archives. (0 = unlimited depth)
- **ArchiveMaxSize**. Maximum file size (in bytes) of an archive that will be scanned. (0 = unlimited size)
- **ArchiveMaxRatio**. Maximum compression ratio of an archive that will be scanned.
- **BlockSuspiciousArchive**. If set to **YES**, email that reach the limits ArchiveMaxRecursion, ArchiveMaxSize or ArchiveMaxRatio are blocked.
- **BlockEncrypedArchive**. If set to **YES**, email with encryped archives are blocked.
- **BlockUnsupportedArchive**. If set to **YES**, email with archives that cannot be scanned are blocked.
- **BlockOnError**. If set to **YES** mail that cannot be scanned due to scan timeout or process error are blocked.

Handling Envelope Recipient Addresses

Source routing is a technique whereby the sender of a packet can specify the route that a packet should take through the network.

The following options concern the source routing used by avgated. For a detailed description, read the AVMailGate documentation.

- **AllowSourceRouting**
- **InEnvelopeAddressesBangIs**
- **InEnvelopeAddressesPercentIs**
- **AcceptLooseDomainName**

Adding a Notification in the Body of Transmitted Mails

- **AddStatusInBody**. If set to **YES**, a default status text is inserted in the email's body.

  If you want to insert your own text, you can specify a file name here. For no status text, enter **NO** (default).

- **MaxMessageSizeStatus**. Specify a message size up to where the status text is added to the message.

  Syntax: **MaxMessageSizeStatus *X*m | k | b**

- **ForwardAllEmailAsMIME**. If set to **YES**, all incoming non-MIME mails are converted to MIME.

- **AddPrecedenceHeader**. If set to **YES**, each notice mail is maked with "Precedence:" in the header. You also can enter a custom text.

- **AddressFilter**. If set to **YES** each sender and/or recipient address will be matched against the tables /etc/avmailgate.scan and /etc/avmailgate.ignore.

  The order in which the tables are scaned can be specified in **FilterTableOrder**. (first hit matches)

- **UseProxy**. You can optimize the scans by using the proxy feature in an AVMailGate pool.

  The number of anti-virus scanners in the pool can be specified with the **ProxyScanners** option.

  ProxyConnections specifies the number of simultaneous allowed connections.

- **AddHeaderToNotice**. If set to **YES**, a mail header is added to postmaster notice mails.

- **BounceMessageSizeBody**, **BounceMessageSizeHeader**. Limit the size (in bytes) of bounced mails.

- **AddXHeaderInfo**. If set to **YES**, the information about the scanning status is added to the header of the checked mail.

- **AddReceivedByHeader**. If set to **YES**, a “Received:” is added to the mail’s header.
- **MaxHopCount**. If there are more than **MaxHopCount** "Received:" lines in the header, the mail will not be accepted.

  Prevent mail loops.

Other Configuration Files

There are four more configuration files for AVMailGate. These files contain a couple of regular expressions.We will only give a short overview here:

- **/etc/avmailgate.acl**. Defines which hosts are considered local and for which the server is allowed to relay emails.
- **/etc/avmailgate.ignore**. Defines mail addresses that should not be scanned.
- **/etc/avmailgate.scan**. Defines mail addresses that are always scanned.
- **/etc/avmailgate.warn**. In this file one can specify who receives a mail in case of an alert.

Update AVMailGate

The file that includes the virus signatures is /usr/lib/AntiVir/antivir[0123].vdf.

To update these files, enter

/usr/lib/AntiVir/antivir --update

The output looks like the following:

```
AntiVir / Linux Version 2.1.5-24 +gui
Copyright (c) 1994-2005 by H+BEDV Datentechnik GmbH.
All rights reserved.


Warning: the file "antivir.vdf" is more than 14 days old
checking for updates

06.32.00.60   =  06.32.00.60  [vdf database (part 0), on-disk]
06.32.18.16   <  06.34.00.105 [vdf database (part 1), on-disk]
06.32.18.17   <  06.34.00.106 [vdf database (part 2), on-disk]
06.33.00.07   <  06.34.00.124 [vdf database (part 3), on-disk]
06.33.00.11   <  06.34.00.14  [scan engine, running]
06.33.00.11   <  06.34.00.14  [scan engine, on-disk]
antivir1.vdf 100% |*************************| 1630 KB    1.59 MB/s   0:00
ETA
antivir2.vdf 100% |*************************|    1 KB    0.00 KB/s  --:--
ETA
antivir3.vdf 100% |*************************|   35 KB    0.00 KB/s  --:--
ETA
antivir 100% |******************************|  695 KB    0.00 KB/s  --:--
ETA
06.32.00.60   =  06.32.00.60  [vdf database (part 0), on-disk]
06.34.00.105  =  06.34.00.105 [vdf database (part 1), on-disk]
06.34.00.106  =  06.34.00.106 [vdf database (part 2), on-disk]
06.34.00.124  =  06.34.00.124 [vdf database (part 3), on-disk]
06.34.00.14   =  06.34.00.14  [scan engine, on-disk]

scan engine  06.33.00.11 --> 06.34.00.14 (/usr/lib/AntiVir/antivir)
vdf database 06.33.00.07 --> 06.34.00.124 (/usr/lib/AntiVir/antivir1.vdf,
/usr/lib/AntiVir/antivir2.vdf, /usr/lib/AntiVir/antivir3.vdf)

AntiVir updated successfully
```

If you only want to check whether new updates are available without updating the files, enter

/usr/lib/AntiVir/antivir --update --check

The output looks like the following:

```
AntiVir / Linux Version 2.1.5-24 +gui
Copyright (c) 1994-2005 by H+BEDV Datentechnik GmbH.
All rights reserved.

checking for updates

06.32.00.60   =  06.32.00.60  [vdf database (part 0), on-disk]
06.32.18.16   <  06.34.00.105 [vdf database (part 1), on-disk]
06.32.18.17   <  06.34.00.106 [vdf database (part 2), on-disk]
06.33.00.07   <  06.34.00.124 [vdf database (part 3), on-disk]
06.33.00.11   <  06.34.00.14  [scan engine, running]
06.33.00.11   <  06.34.00.14  [scan engine, on-disk]

an update for the scan engine is available (/usr/lib/AntiVir/antivir)
an update for the VDF database is available
(/usr/lib/AntiVir/antivir1.vdf, /usr/lib/AntiVir/antivir2.vdf,
/usr/lib/AntiVir/antivir3.vdf)
```

Exercise 4-9 Use AVMailGate as a Virus Scanner for Email

In this exercise, you install and configure AVMailGate as a virus scanner for mails.

You will find this exercise in the workbook.

(End of Exercise)

AMaViSd-new

AMaViSd-new (*A Mail Virus Scanner*) consists of the daemon and some optional helper programs, which are only needed during setup between the message transfer agent (MTA) and one or more content checkers (virus scanners or SpamAssassin).

The mail server sends all incoming and outgoing mails to AMaViSd. The email will be extracted and tested by AMaViSd-new.

AMaViSd recognizes four kinds of unwanted mails. They are mails that have:

1. Invalid headers
2. Banned file types
3. Viruses
4. Spam

AMaViSd tests incoming mails for these four types in the order listed.

If nothing bad is found, AMaViSd-new will send the email back to the mail server, ready for delivery.

By default, AMaViSd-new listens at port 10024 for incoming email from the mail server.

AMaViSd-new sends clean mails to the mail server via port 10025. It is normally positioned at or near a central mail server, not necessarily where users' mailboxes and final delivery takes place.

This topic describes the following:

- Install AMaViSd-new
- Configure AMaViSd-new

Install AMaViSd-new

To install AMaViSd-new, select the package **amavisd-new**.

AMaViSd-new will not work if no virus scanner is installed on your system. If you want to use AMaViSd-new only for spam filtering, search for **@bypass_virus_checks_acl** in /etc/amavisd.conf and remove the comment sign ("#") at the beginning of the line.

The compressing tools gzip, bzip2, arc, lha, unrar, zoo, cpio, and lzop should be installed, too.

During the SUSE Linux Enterprise Server 10 installation, a user vscan is created. It is used for AMaViSd.

```
da51:~ # grep vscan /etc/passwd
vscan:x:65:104:Vscan account:/var/spool/amavis:/bin/false
```

After installing AMaViSd-new, you can start the daemon with **rcamavis start**.

The daemon should listen on port 10024.

```
da51:~ # telnet 127.0.0.1 10024
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 [127.0.0.1] ESMTP amavisd-new service ready
```


Using SMTP commands you can write an email now.

```
da51:~ # telnet 127.0.0.1 10024
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 [127.0.0.1] ESMTP amavisd-new service ready
mail from: jgoldman@digitalairlines.com
250 2.1.0 Sender jgoldman@digitalairlines.com OK
rcpt to: geeko@digitalairlines.com
250 2.1.5 Recipient geeko@digitalairlines.com OK
data
354 End data with <CR><LF>.<CR><LF>
Subject: Test

Test
.
250 2.6.0 Ok, id=09232-01, from MTA([127.0.0.1]:10025): 250
Ok: queued as 5747B16A68
quit
221 2.0.0 [127.0.0.1] amavisd-new closing transmission
channel
Connection closed by foreign host.
da51:~ #
```

The email header should contain a line for amavisd.

```
geeko@da51:~> mail
mailx version nail 11.25 7/29/05.  Type ? for help.
"/var/spool/mail/geeko": 1 message 1 new
>N  1 jgoldman@digitalai Tue May 16 15:30   20/829   Test
? 1
Message  1:
From jgoldman@digitalairlines.com  Tue May 16 15:30:58 2006
X-Original-To: geeko@digitalairlines.com
Delivered-To: geeko@digitalairlines.com
Subject: Test
X-Virus-Scanned: amavisd-new at example.com
Date: Tue, 16 May 2006 15:30:58 -0400 (EDT)
From: jgoldman@digitalairlines.com
To: undisclosed-recipients:;

Test

?
```

Configure AMaViSd-new

On SUSE Linux Enterprise Server 10 you can configure AMaViSd-new by editing one of the following files:

- /etc/sysconfig/amavis
- /etc/amavisd.conf

/etc/sysconfig/amavis

The configuration file /etc/sysconfig/amavis is only available on SUSE Linux products.

In this file there are only two parameters:

- **USE_AMAVIS**. If set to **yes**, Sendmail or Postfix are prepared to use AMaViSd-new.

- **AMAVIS_SENDMAIL_MILTER**. If set to **yes**, the milter (***Mail Filter***) interface of Sendmail will be started.

  Using the milter interface, it is possible to connect mail filter applications to Sendmail in a standarized form.

After changing the file /etc/sysconfig/amavis, you have to run the command **SuSEconfig**.

Setting USE_AMAVIS to yes makes three changes in /etc/postfix/master.cf:

- The smtp protocol

```
smtp      inet  n       -       n       -       2       smtpd -o
content_filter=smtp:[127.0.0.1]:10024
```

- The smtps protocol (still marked as comment)

```
#smtps    inet  n       -       n       -       2       smtpd -o
smtpd_tls_wrappermode=yes -o content_filter=smtp:[127.0.0.1]:10024
```

- Port 10025

```
localhost:10025 inet    n       -       n       -       -       smtpd -o
content_filter=
```

In this file you also have to add a process for amavis:

```
smtp-amavis unix -      -       n       -       2  smtp
    -o smtp_data_done_timeout=1200
    -o smtp_send_xforward_command=yes
    -o disable_dns_lookups=yes
    -o max_use=20
```

In the Postfix configuration file /etc/postfix/main.cf you have to add the following line:

```
content_filter = smtp-amavis:[127.0.0.1]:10024
```

Restart Postfix by entering **rcpostfix reload**.

/etc/amavisd.conf

AMaViSd is is configured by editing the file /etc/amavisd.conf. The syntax of this configuration file is plain Perl code. Because of this, each line ends in a semicolon.

In Perl strings in double quotation marks can include variables that start with “$” or “@”. To include characters “@” and “$” in double-quoted strings, they must be preceded by a backslash.

In single-quoted strings, the “$” and “@” lose their special meaning, so it is usually easier to use single quoted strings.

In both cases, the backslash needs to be doubled.

In the documentation directory of AMaViSd-new (/usr/share/doc/packages/amavisd-new/), two more variants of the configuration file are available:

- **amavisd.conf-default**. Includes all possible parameters with their defaults.
- **amavisd.conf-sample**. A more structured file with a lot of explanations and examples.

In this section we want to discuss some of the most important parameters in order of occurence in the /etc/amavisd.conf file.

- **@bypass_virus_checks_maps**. This array includes a list of virus lookup tables. You can disable virus checking by setting this array to “1” (uncomment the line).

- **@bypass_spam_checks_maps**. This array includes a list of spam lookup tables. You can disable spam checking by setting this array to "1" (uncomment the line).

Next to @bypass_virus_checks_maps and @bypass_spam_checks_maps, there are two more arrays of the same kind available:
@bypass_banned_checks_maps enables checking for banned names or file types.
@bypass_header_checks_maps enables checking for invalid headers.

- **$max_servers.** Number of pre-forked children.

  Should match the number of your MTA pipe, e.g., the **maxproc** field in /etc/postfix/master.cf.

```
#
==========================================================================
=
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (100)
#
==========================================================================
=
smtp      inet  n       -       n       -       2       smtpd -o
content_filter=smtp:[127.0.0.1]:10024
```

- **$daemon_user** and **$daemon_group**. User and group to which the daemon will change.
- **$mydomain**. The domain name.
- **$MYHOME**. Location where the AMaViSd-new files are stored.
- **$TEMPBASE**. The path of a temporary directory that can be used by AMaViSd-new. This directory must exist or needs to be created manually.

  In the following line an environment variable **$TMPDIR** is defined with the content of the $TEMPBASE variable.

- **$QUARANTINEDIR**. The path to where infected mails can be put. This can be
  - ***a file***. Path does not end with backslash.
  - ***a directory***. Path ends with backslash.
  - **disabled**. Leave it empty.
- **$quarantine_subdir_levels**. If set to “1” subdirectories in $QUARANTINEDIR are created to disperse quarantine.
- **$daemon_chroot_dir**. Run the daemon in the specified chroot jail. If you do not want chroot, leave it empty.
- **$db_home**. Path of the databases. Default: $MYHOME/db.
- **$helpers_home**. Sets the environment variable $HOME. The value is passed to other SpamAssassin modules.
- **$pid_file**. Path of the PID file.
- **$lock_file**. Path of the lock file.
- **@local_domains_maps**. List of lookup tables that can be used to decide whether a recipient is local or not, i.e., if the message is outgoing or not.
- **@mynetworks**. List of IP ranges which determines if the original SMTP client IP address belongs to the internal networks, i.e. if mail is coming from inside.
- **$log_level**. Log level.
  - **0**. Startup/exit/failure messages, viruses detected
  - **1**. Args passed from client, some more interesting messages
  - **2**. Virus scanner output, timing
  - **3**. Server, client
  - **4**. Decompose parts
  - **5**. More debug details
- **$log_recip_templ**. Template for log file entries.

A list of the available macros can be found in /usr/share/doc/packages/amavisd-new/README_FILES/README.customize.

- **$DO_SYSLOG**. If set to “1”, the syslog daemon is used for loggings.
- **$SYSLOG_LEVEL**. Level of syslog loggings.
- **$enable_db**. Enable use of BerkeleyDB/libdb. If it is enabled, you can also enable the libdb cache using **$enable_global_cache**.
- **$inet_socket_port**. Port number that the AMaViSd-new should listen to.
- **$unix_socketname**. If you are using Sendmail milter, you have to enter the socket name here.
- **$sa_tag_level_deflt**. Spam info headers are added to the mail if the spam level is at or above the given number.
- **$sa_tag2_level_deflt**. Spam detected headers are added to the mail if the spam level is at or above the given number.
- **$sa_kill_level_deflt**. Spam evasive actions (bounce/reject/drop) are triggered if the spam level is at or above the given number.
- **$sa_dsn_cutoff_level**. Spam with a spam level beyond this number is not sent.
- **$sa_quarantine_cutoff_level**. Spam with a spam level beyond this number is not quarantined.
- **$sa_mail_body_size_limit**. Email messages larger than the given number is not passed to SpamAssassin. (Less than 1% of spam is larger than 64 KB.)
- **$sa_local_tests_only**. If set to “1”, no SpamAssassin tests requiring Internet access are performed.

- **$sa_auto_whitelist**. If set to “1”, AWL (auto-whitelist) in SpamAssassin 2.63 or older is turned on (irrelevant for SpamAssassin 3.0; on SUSE Linux Enterprise Server 10, version 3.1.0 is available)

AWL tracks scores for your regular correspondents in a small on-disk database. Since version 3.0, it is enabled by default.

- **@lookup_sql_dsn**. Array with information about where to find SQL server(s) and database to support SQL lookups. One item includes a triple of data: source name, user, and password.
- **$virus_admin**. Fully qualified address of the antivirus administrator.
- **$mailfrom_notify_admin**. Fully qualified address of the sender of admin notifications.
- **$mailfrom_notify_recip**. Fully qualified address of the sender of virus notifications.
- **$mailfrom_notify_spamadmin**. Fully qualified address of the sender of spam notifications.
- **$mailfrom_to_quarantine**. Whom quarantined messages appear to be sent from. If undefined, the original sender is used.
- **@addr_extension_virus_maps**. The specified string is added to the recipient’s address if a virus is detected.

  The **@addr_extension_spam_maps**, **@addr_extension_banned_maps**, and **@addr_extension_bad_header_maps** strings work in the same way.

  The string is separated from the recipient address by the string specified in **$recipient_delimiter**.

  Example:
  geeko@digitalairlines.com > geeko+spam@digitalairlines.com

- **$path**. The content of this variable is passed to the PATH environment variable.
- **$dspam**. Activate the dspam content filter (http://www.nuclearelephant.com/projects/dspam/).
- **$MAXLEVELS**. Maximum recursion level for extraction and decoding.
- **$MAXFILES**. Maximum number of extracted files.
- **$MIN_EXPANSION_QUOTA**, **$MAX_EXPANSION_QUOTA**. Minimum and maximum storage size (in bytes) that is available for mail extraction.
- **$sa_spam_subject_tag**. String that is prepended to the subject header when message exceeds **$sa_tag2_level_deflt** level.
- **@*_lovers_maps**. Email to the specified recipients is not examined and filtered.
- **@blacklist_sender_maps**. A message from a blacklisted envelope sender address is marked as spam.

  A **@whitelist_sender_maps** is also available. Mail with sender addresses from this array are delivered although they are marked as spam.
- **@score_sender_maps**. Using this variable you can add or substract a specified value to/from the spam value.

The next variables (beginning with **@viruses_that_fake_sender_maps**) contain regular expressions to filter mails.

Many regular expressions are predefined and you can enable them by removing the comment hash sign at the beginning of the line(s).

Of course you can modify the regular expressions if they do not fit your needs.

Some other important variables are:

- **$final_virus_destiny**, **$final_banned_destiny**, **$final_spam_destiny**, **$final_bad_header_destiny**. Defines what to do with email that has a virus, a banned sender, spam content, or incorrect headers.

  You can use the following values:

  - **D_PASS**. Mail will pass to recipients, regardless of bad contents.
  - **D_DISCARD**. Mail will not be delivered to its recipients; sender will not be notified.
  - **D_BOUNCE**. Mail will not be delivered to its recipients; a nondelivery notification (bounce) will be sent to the sender by AMaViSd-new.
  - **D_REJECT**. Mail will not be delivered to its recipients; the sender should preferably get a rejection notification, (SMTP permanent reject response or nondelivery notification from the MTA).

    If this is not possible, AMaViSd-new sends a bounce by itself (the same as **D_BOUNCE**).

- **$notify_method**. Specify a host and port where the notifications are sent to.
- **$notify_sender_templ**. Add this variable if you do not want the sender of an email notified.
- **$notify_virus_sender_templ**. Specify a text file if you do not want the default notification sent to the sender of an email that contained a virus.
- **$notify_virus_admin_templ**. Specify a text file if you do not want the administrator notified when a virus is detected.
- **$notify_virus_recips_templ**. Specify a text file if you do not want to notify the recipients of an email that contained a virus.
- **$notify_spam_sender_templ**. Specify a text file if you do not want to notify the sender of an email that contained spam.

- **$notify_spam_admin_templ**. Specify a text file if you do not want to notify the administrator if a spam email is detected.

Exercise 4-10 Use AMaViSd as Virus Scanner for Email

In this exercise, you install and configure AMaViSd.

You will find this exercise in the workbook.

(End of Exercise)

Summary

Objective	Summary
1. Understand the Function of the Three Mail Agents	Emails are passed by the user's email software to the MTA. The MTA sends them to a recipient MTA. The MTA specifies which local user the received message is intended for and passes this to an MDA, which stores it in the right location. The MUA reads saved messages and passes new messages to the MTA. The MUA is the interface between the MDA and the user.

Objective	Summary
2. Use the Basic Linux mail Command	Some programs write status mail messages or notes to the user root. If you log in at a virtual terminal, you are notified if there are messages waiting for you. You can use the **mail** command to read the messages. You can also use the mail program as a mail client on the command line. To finish the text body, you have to enter a single full stop in the last line. The following SMTP commands must be implemented in an SMTP server to provide SMTP communication: ■ **HELO** ■ **MAIL FROM:** ■ **RCPT TO:** ■ **DATA** ■ **RSET** ■ **NOOP** ■ **QUIT**

Objective	Summary
3. Understand the Architecture and Components of Postfix	An email sent locally is placed by Postfix's Sendmail via the postdrop command into the maildrop queue. Then it is picked up by the pickup daemon, which checks it for ■ Content ■ Size ■ Other factors based on rules Then the cleanup daemon ■ Inserts missing header lines ■ Deletes double recipient addresses ■ Converts the email address in the header via the trivial-rewrite daemon to the user@fully-qualified-domain convention ■ Writes data in the header according to the rules in the canonical and virtual tables After this, the email is copied to the incoming queue and the queue manager is informed of the arrival of this email.

Objective	Summary
3. Understand the Architecture and Components of Postfix (*continued*)	Email received over the Internet or the LAN is accepted by the daemon smtpd. Emails are checked for ■ Content ■ Size ■ Other factors before passing it to the cleanup daemon. The queue manager fetches an email placed in the incoming queue and copies the email to the active queue, assuming it is empty. The trivial-rewrite daemon, using the transport table, checks to see if the recipient of the email is on the local system or on a remote system. Important components of Postfix are in: ■ **/etc/aliases**. Contains local address aliases. ■ **/etc/postfix/**. All the configuration files defining Postfix mail processing are located in this directory. ■ **/usr/lib/postfix/**. This directory contains all the programs needed by Postfix.

Objective	Summary
3. Understand the Architecture and Components of Postfix (*continued*)	■ **/usr/sbin/**. This directory contains the administration programs for maintaining and manually controlling Postfix. ■ **/usr/bin/**. Symbolic links with the names mailq and newaliases are found here. ■ **/var/spool/postfix/**. This directory contains the queue directories for Postfix and its own directory etc/ and lib/ for Postfix processes that run in a chroot environment. ■ **/usr/share/man/man[1|5|8]/**. These directories contain the manual pages. ■ **/usr/share/doc/packages/postfix/** The documentation for Postfix can be found here.
4. Start, Stop, and Reinitialize Postfix	Use **rcpostfix start** to start the Postfix daemon. Use **rcpostfix stop** to stop the Postfix daemon.

Objective	Summary
5. Configure Postfix	The Postfix master daemon is configured via the file **/etc/postfix/master.cf** Each line in the file contains an entry for one Postfix process. The individual columns have the following meanings: ■ **service**. The name of the Postfix process. ■ **type**. Allows a connection type to be given. ■ **private**. Configures access to the service. ■ **unpriv**. Configures the UID under which this service is running. ■ **chroot**. Specifies the chroot behavior of the service. ■ **wakeup**. Runs the service again after the given number of seconds have expired. ■ **maxproc**. Defines the maximum number of processes that can be run simultaneously. ■ **command + args**. Configures the command to be carried out, including the required arguments.

Objective	Summary
5. Configure Postfix (*continued*)	All further configuration definitions (apart from the configuration of processing rules in lookup tables) are set in the file **/etc/postfix/main.cf** On SUSE Linux Enterprise Server 10, the most common parameters of this file can be modified using variables in the files **/etc/sysconfig/mail** and **/etc/sysconfig/postfix** For the MTA to operate correctly, you need to make two settings in the file /etc/sysconfig/mail: ■ Enter the FQDN in the variable FROM_HEADER. ■ Set the variable SMTPD_LISTEN_REMOTE to yes so Postfix will listen on port 25 for arriving email. Otherwise only email from the local host will be accepted. Modifications in the file /etc/sysconfig/postfix are only adopted in the file /etc/postfix/main.cf and, in some cases, in the file /etc/postfix/master.cf after executing /sbin/SuSEconfig.

Objective	Summary
5. Configure Postfix (*continued*)	A mail server that forwards email to a provider's mail server has to ■ Accept mail from the intranet clients. ■ Reject mail delivered by other clients. ■ Possibly rewrite sender addresses. ■ Submit all mail to the provider's mail server. In addition, a mail server that also receives mail over the Internet has to ■ Accept mail that comes from the Internet and is addressed to your domain. ■ Reject mail that comes from the Internet and is not addressed to your domain. ■ Reject mail from known spam sources. Regardless of the individual configuration of Postfix in the last case, the server must be introduced to DNS as the responsible mail server by means of an MX record.

Objective	Summary
5. Configure Postfix (*continued*)	Lookup tables contain rules for processing email within the overall Postfix system. These tables are activated by variables in the file **/etc/postfix/main.cf** which is then defined as **/etc/postfix/*lookup-table*** After a lookup table has been defined, it needs to be converted to the required format (usually in the form of a hash table) using the command **postmap**. Use the access lookup table to reject or allow email from defined senders. Use the canonical lookup table to rewrite sender and recipient addresses of incoming and outgoing email. Use the relocated lookup table to return the corresponding bounce email, with a note to senders of email for users that no longer exist on this system. Use the transport lookup table to define email routing for special email address ranges.

Objective	Summary
5. Configure Postfix (*continued*)	Use the virtual lookup table to set up email for a number of domains with separate user names. Use the aliases lookup table to define email aliases for delivering email.

Objective	Summary
6. Use Postfix Tools	The following are Postfix tools: ■ **newaliases**. Converts the ASCII file /etc/aliases to the hash table /etc/aliases.db. ■ **mailq**. Lists all email in the mail queues that have not yet been sent. ■ **postalias**. Converts the ASCII file /etc/aliases to the hash table /etc/aliases.db. Same as **newaliases**. ■ **postcat**. Displays the contents of a file from the queue directories in a readable form. ■ **postconf**. Displays the values of all variables. Enter **postconf -e key=value** to modify variables directly. These changes are automatically integrated in the file main.cf. ■ **postdrop**. This is run automatically by the command sendmail. ■ **postfix**. Enables configuration errors to be found, forces email from the deferred queue to be delivered immediately, or rereads the Postfix configuration files. ■ **postmap**. Generates the hash tables for the lookup tables in the directory /etc/postfix/. ■ **postsuper**. Removes all files that are not normal files or directories.

Objective	Summary
7. Receive Email over IMAP and POP3	Cyrus IMAPd is designed for servers without user accounts for normal users. ■ **/etc/cyrus.conf**. The basic configuration file of Cyrus IMAPd. ■ **/etc/imapd.conf**. The configuration file of IMAPd. ■ **/var/lib/imap/mailboxes.db**. Includes important information about the mailboxes. ■ **/var/lib/imap/log/**. Stores log information. ■ **/var/lib/imap/quota/**. Defines the quotas for each IMAP account. ■ **/var/lib/imap/shutdown**. If this file exists, the IMAP server sends the first line of this file as an alert to the client. ■ **/var/lib/imap/msg/motd**. If this file exists, the IMAP server sends the first line to the client after the connection is created. As soon as QPopper is installed, it is available for retrieving mail via the POP3 protocol, if the superdaemon xinetd was configured and started in the system.

Objective	Summary
7. Receive Email over IMAP and POP3 (*continued*)	Procmail is a highly flexible MDA. It can sort email into mailboxes, forward them to other recipients, or delete them according to almost any arbitrary criteria a user specifies. If Postfix is used as the MTA, an entry is required in /etc/postfix/main.cf. The most interesting aspect of using Procmail is the prefiltering and distribution of email based on the information in the mail header or the email body. Each email recipient can create a file .procmailrc in his or her home directory and install his or her own desired settings for Procmail. Although it is possible to create a system-wide configuration, using the configuration file /etc/procmailrc.

Objective	Summary
8. Manage Spam	The most common spam-filtering tool with Linux is SpamAssassin. It tests the email body, header, URI, and the attachments. It also uses black lists and white lists of other organizations. Select the package spamassassin to install the spam filter. After the installation, the following executable files are available: ■ **spamassassin**. Starts SpamAssassin. ■ **spamd**. A daemonized version of the SpamAssassin executable. This daemon is useful for automated mail checks. ■ **spamc**. A client for spamd. Use spamc instead of SpamAssassin in your own scripts. ■ **sa-learn**. A tool to train SpamAssassin. SpamAssassin is configured by editing the file **/etc/mail/spamassassin/local.cf**

Objective	Summary
9. Use a Virus Scanner for Email	AVMailGate is an antivirus email filter from H+BEDV Datentechnik GmbH. AVMailGate can update the virus definition file and the engine itself. AVMailGate is composed of two kinds of processes: ■ **avgated**. The smtpd receiver that stores incoming email in the input queue. ■ **avgatefwd**. Virus scanning function and SMTP forwarder and sendmail invoker. Both processes can be configured by editing the file ***/etc/avmailgate.conf*** Clam AntiVirus is an antivirus toolkit for UNIX. The main purpose of this software is integration with mail servers (attachment scanning). The package provides ■ A flexible and scalable multithreaded daemon ■ A command line scanner ■ A tool for automatic updating via Internet The programs are based on a shared library distributed with the Clam AntiVirus package, which you can use with your own software.

Objective	Summary
9. Use a Virus Scanner for Email (continued)	AMaViSd-new consists of ■ The daemon. ■ Optional helper programs that are only needed in setup between the message transfer agent (MTA). ■ One or more content checkers: virus scanners such as SpamAssassin. The email server sends all incoming and outgoing email to AMaViSd. The emails will be extracted by AMaViSd-new and tested with a virus scanner. If no viruses are found, AMaViSd-new sends the email back to the mail server ready for delivery. AMaViSd-new is configured in the file **/etc/amavisd.conf**

SECTION 5 Use OpenSLP

In this section, you learn what OpenSLP is, how to configure it, and how to get information on services using OpenSLP.

The OpenSLP project home page is at

http://openslp.org

Novell, Inc. is the current maintainer of the code base.

Objectives

1. Understand what OpenSLP Is
2. Understand the SLP Agents
3. Configure OpenSLP
4. Use SLP Frontends

Objective 1 Understand what OpenSLP Is

Traditionally, you access a service on an IP network by using the host IP address. That process can be simplified with DNS. However, these methods are static and do not guarantee that the host or the service on that host is actually available.

The Service Location Protocol (SLP) provides the dynamic discovery of services. Only active services are visible through SLP.

You only need to know the description of the desired service. Based on this description, SLP can return the URL of the desired service.

For example, a search for NTP will result in addresses for all active NTP services on the network.

SLP was originally an Internet Engineering Task Force (IETF) protocol that provides a framework to allow networking applications to discover

- The existence
- The location
- The configuration

of networked services in enterprise networks.

OpenSLP is available on SUSE Linux Enterprise Server 10.

OpenSLP is an Open Source implementation of the SLP version 2 protocol as defined by RFC 2608 and RFC 2614. OpenSLP registers information on active services on the network in a cached database and allows clients to query the database to find these services.

From a high-level viewpoint, the information that OpenSLP maintains about a service is simply the service name and the IP address of the host that is running this service.

On SUSE Linux Enterprise Server 10, many applications including the following, have already integrated OpenSLP support:

- CUPS
- ypserv (NIS)
- OpenLDAP2
- SANE
- Postfix
- SSH (via fish)
- NTP

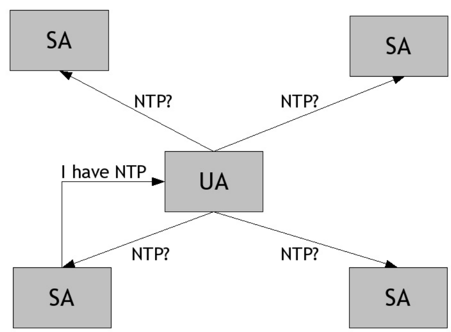
SLP is described in great detail in the following RFCs:
- RFC 2608, "Service Location Protocol, Version 2"
- RFC 2609, "Service Templates and Service Schemes"
- RPC 2610, "DHCP via SLP"
- RFC 2614, "An API for Service Location Protocol"

For more information on SLP, refer to the home page of the OpenSLP project at http://www.openslp.org, or see your SUSE Linux Enterprise Server 10 system at file:/usr/share/doc/packages/openslp/*.

Objective 2 Understand the SLP Agents

SLP uses agents to represent applications and services on the network.

Figure 5-1



An agent is a software entity that processes SLP protocol messages.

There are three types of SLP agents:

- User Agent
- Service Agent
- Directory Agent

These agents communicate with each other using different types of Messages.

To understand how these messages are sent and received, you need to know the differences between Unicast, Multicast, and Broadcast Protocols.

User Agent

The SLP User Agent (UA) is software that looks for the location of one or more hosts running specified services.

Usually implemented (at least partially) as a library to which client applications link, it provides client applications with a simple interface for accessing SLP registered service information.

User Agents use two possible methods to identify the services they require:

- If Directory Agents ***are used*** on the network, the User Agent will send unicast requests to the Directory Agents servicing the specified services.
- If Directory Agents ***are not used*** on the network, the User Agent will send out a multicast request for a specified service.

Service Agent

The SLP Service Agent (SA) is software that advertises the location of one or more active services.

SLP advertisement is designed to be both scalable and effective, minimizing the use of network bandwidth through the use of targeted multicast messages and unicast responses to queries.

Service Agents use two possible methods to advertise the services they have registered:

- If Directory Agents ***are used*** on the network, the Service Agent sends service registrations to the Directory Agent.
- If Directory Agents ***are not used*** on the network, the Service Agent answers requests for services that match the services that are registered.

  Once this registration has been performed, requests for services can be serviced by the Directory Agent instead.

All service requests and replies will be unicast packages.

The Service Agent will stay passive until the service terminates or the life time is reached. Only then will it contact the Directory Agent again.

Directory Agent

The SLP Directory Agent (DA) is software that acts as a centralized repository for service location information.

Directory Agents receive registration requests from Service Agents and use the service information to answer requests for services from User Agents.

Directory Agents can be implemented in a large environment to reduce traffic and segment service availability to manually defined scopes.

Both Service Agents and User Agents make it a priority to discover available Directory Agents, because using a Directory Agent minimizes the number of multicast messages sent by the protocol on the network.

The Service Agent registers itself with the Directory Agent multicast group so that User Agents can locate the Directory Agent and become able to locate active services on the network using unicast packages only.

Messages

In order to provide a framework for service location, SLP agents communicate with each other using 11 different types of messages.

The dialog between agents is usually limited to very simple exchanges of request and reply messages:

- **Service Request (SrvRqst)**. Message sent by User Agents to Service Agents or Directory Agents to request the location of a service.
- **Service Reply (SrvRply)**. Message sent by Service Agents or Directory Agents in response to a SrvRqst message.

  The SrvRply contains the URL of the requested service.
- **Service Registration (SrvReg)**. Message sent by Service Agents to Directory Agents containing information about a service that is available.
- **Service Deregister (SrvDeReg)**. Message sent by Service Agents to inform Directory Agents that a service is no longer available.

  The service informs the Service Agent to deregister itself immediately before it terminates.
- **Service Acknowledge (SrvAck)**. A generic acknowledgment that is sent by Directory Agents to Service Agents in response to SrvReg and SrvDeReg messages.
- **Attribute Request (AttrRqst)**. Message sent by User Agents to request the attributes of a service.
- **Attribute Reply (AttrRply)**. Message sent by Service Agents or Directory Agents in response to an AttrRqst.

  The AttrRply contains the list of attributes that were requested.
- **Service Type Request (SrvTypeRqst)**. Message sent by User Agents to Service Agents or Directory Agents requesting the types of services that are available.
- **Service Type Reply (SrvTypeRply)**. Message by Service Agents or Directory Agents in response to a SrvTypeRqst.

  The SrvTypeRply contains a list of requested service types.
- **DA Advertisement (DAAdvert)**. Message sent by Directory Agents to let Service Agents and User Agents know where they are.

- **SA Advertisement (SAAdvert)**. Message sent by Service Agents to let User Agents know where they are.

Unicast, Multicast, and Broadcast Protocols

SLP is a unicast and multicast protocol.

This means that the messages described above can be sent

- To one agent at a time (unicast)

  or

- To all agents (that are listening) at the same time (multicast)

A multicast is not a broadcast.

In theory, broadcast messages are heard by every node on the network, but multicast messages are only heard by the nodes on the network that have joined the multicast group.

Network routers filter almost all broadcast and multicast traffic.

This means that broadcasts and multicasts generated on one subnet will not be forwarded or routed to any of the other subnets connected to the router. (From the router's perspective, a subnet comprises all machines connected to one of its ports.)

By default, broadcast and multicast are not forwarded by routers.

You can either change your router configuration or use Directory Agents in every network.

The Directory Agents can then share their service information using unicast (routable) packages, and all service information is available in all networks.

In this course, we will concentrate on the User Agent/Service Agent communication only.

OpenSLP needs a mulitcast route and uses well-known SLP ports:

```
da51:~ # grep slp /etc/services
eicon-slp       1440/tcp   # Eicon Service Location Protocol
eicon-slp       1440/udp   # Eicon Service Location Protocol
slp             1605/tcp   # Salutation Manager (Salutation Protocol)
slp             1605/udp   # Salutation Manager (Salutation Protocol)
sslp            1750/tcp    # Simple Socket Library's PortMaster
sslp            1750/udp    # Simple Socket Library's PortMaster
slp-notify      1847/tcp    # SLP Notification
slp-notify      1847/udp    # SLP Notification
csvr-sslproxy   3191/tcp   # ConServR SSL Proxy
csvr-sslproxy   3191/udp   # ConServR SSL Proxy
epl-slp         3819/tcp   # EPL Sequ Layer Protocol
epl-slp         3819/udp   # EPL Sequ Layer Protocol
ds-slp          4406/tcp   # ASIGRA Televaulting DS-Sleeper Service
ds-slp          4406/udp   # ASIGRA Televaulting DS-Sleeper Service
da51:~ #
```

Objective 3 Configure OpenSLP

On SUSE Linux Enterprise Server 10, the following packages are installed by default:

- **openslp-server**. This package contains the SLP server (daemon), the init script, and documentation.

  Every system that provides any service that should be used via an SLP client has to run this server.

- **openslp**. This package includes runtime libraries and slptool, a command line frontend.

To ensure that OpenSLP provides service information on your local network, you need to know the following:

- The OpenSLP Daemon slpd
- The OpenSLP Configuration File
- The OpenSLP Registration Files

The OpenSLP Daemon slpd

The OpenSLP daemon /usr/sbin/slpd provides

- Service Agent and Directory Agent functionality.
- The ability to maintain a consistent state with respect to the locations of other SLP agents on the network.

On SUSE Linux Enterprise Server 10, the daemon is installed and active by default. Its init script is **/etc/init.d/slpd.**

When spld is started, it reads

- The configuration file /etc/slp.conf (see "The OpenSLP Configuration File" on 5-12).
- The registration files of the services that should be maintained by OpenSLP (see "The OpenSLP Registration Files" on 5-18).

If any changes are made to the configuration files or to the registration files, the daemon needs to be restarted for the changes to be effective.

To do this, enter as root

/etc/init.d/slpd restart

or

rcslpd restart

To stop the daemon, enter as root

/etc/init.d/slpd stop

or

rcslpd stop

To start the daemon, enter as root

/etc/init.d/slpd start

or

rcslpd start

To check the daemon's status, enter as root

/etc/init.d/slpd status

or

rcslpd status

The OpenSLP daemon slpd must run as root initially in order to bind to the well-known SLP port. However, slpd will relinquish root privileges and suid() to the daemon user (if it exists).

The OpenSLP Configuration File

The OpenSLP configuration file is **/etc/slp.conf**. The OpenSLP daemon slpd reads this file when it is started or restarted.

This file contains configuration information that affects

- The operation of the OpenSLP daemon (/usr/sbin/slpd)
- Any application that uses the OpenSLP library (/usr/lib/libslp.so)

The default settings of slpd will meet common requirements. Therefore, you do not need to modify this configuration file.

To understand this file, you need to know the following:

- The Syntax of /etc/slp.conf
- Parameters in /etc/slp.conf

The Syntax of /etc/slp.conf

Each parameter in the slp.conf file is in a single line in the format

parameter=value

For example: **net.slp.useScopes=DEFAULT**

Comment lines begin with "#" or ";".

The file syntax is specified in RFC 2614.

The parameters are described in the following section.

Parameters in /etc/slp.conf

The following is a list of parameters that are supported by OpenSLP:

- **net.slp.interfaces**. This parameter is a comma-separated list of interfaces (local IP addresses) on which a Directory Agent or Service Agent should listen for OpenSLP requests.

  This parameter is optional for the User Agent but required for a Service Agent or Directory Agent to operate properly.

  Both IPv4 and IPv6 addresses may be specified.

  While site-local and global IPv6 addresses are allowed, a Directory Agent or Service Agent can only receive IPv6 multicast on link-local addresses.

  By default, OpenSLP uses all interfaces.

- **net.slp.useScopes**. This parameter is a comma-delimited list of strings indicating the only scopes a User Agent or Service Agent is allowed when making requests or registering, or the scopes a Directory Agent must support.

  The default value is DEFAULT. That means all services are registered in that scope.

- **net.slp.DAAddresses**. This parameter allows you to force User Agents and Service Agents to use specific Directory Agents.

  If this setting is not used, dynamic Directory Agent discovery will be used to determine which Directory Agents to use.

  The default is to use dynamic Directory Agent discovery.

- **net.slp.broadcastAddr**. This parameter is a string indicating the broadcast address to use when sending broadcast packets. This parameter is only applicable when the other broadcast configuration variables are set.

  The default value is 255.255.255.255.

- **net.slp.isBroadcastOnly**. This parameter forces broadcasts to be used instead of mulitcast.

  This parameter is seldom necessary since OpenSLP will automatically use broadcast if multicast is unavailable.

  The default value is false.

- **net.slp.passiveDADetection**. A boolean indicating if passive Directory Agent detection should be used.

  The default value is true.

- **net.slp.DAActive DiscoveryInterval**. A 16-bit positive integer giving the number of seconds between Directory Agent active discovery queries.

  The default value is 900 seconds (15 minutes).

  If the property is set to zero, active discovery is turned off.

  This is useful when the available Directory Agents are explicitly restricted to those obtained from DHCP or the net.slp.DAAddresses property.

- **net.slp.multicastTTL**. A positive integer that is less than or equal to 255.

  The default value is 255.

- **net.slp.multicastMaximumWait**. An integer giving the maximum amount of time (in milliseconds) to perform multicast requests.

  The default value is 15000 ms (15 seconds).

- **net.slp.unicastMaximumWait**. An integer giving the maximum amount of time (in milliseconds) to perform unicast requests.

  The default value is 15000 ms (15 seconds).

- **net.slp.randomWaitBound**. An integer giving the maximum value for all random wait parameters.

The default value is 1000 (1 second).

- **net.slp.MTU**. An integer giving the network packet MTU in bytes.

  This is the maximum size of any datagram, but the implementation might receive a larger datagram.

  The default value is 1400 (bytes).

- **net.slp.securityEnabled**. A boolean indicating whether the agent should enable security for URLs, attribute lists, DAAdverts, and SAAdverts.

  Each agent is responsible for interpreting the property appropriately.

  The default value is false.

- **net.slp.locale**. An RFC 1766 Language Tag [6] for the language locale.

  Setting this property causes the property value to become the default locale for OpenSLP messages.

  This property is also used for Service Agent and Directory Agent configuration.

  The default value is en (English).

- **net.slp.maxResults**. A 32-bit integer giving
  - The maximum number of results to accumulate and return for a synchronous request before the timeout.

    or
  - The maximum number of results to return through a callback if the request results are reported asynchronously.

- **net.slp.isDA**. A boolean indicating whether the OpenSLP server is to act as a Directory Agent.

  If the value is false, the OpenSLP server does not run as a Directory Agent.

The default value is false.

- **net.slp.DaHeartBeat**. A 32-bit integer giving the number of seconds for the Directory Agent heartbeat.

  The default value is 3 hours (10,800 seconds).

  This option will be ignored if the value of net.slp.isDA is false.

- **net.slp.useIPV4**. This parameter specifies whether IPv4 should be used for OpenSLP.

  The default value is true.

- **net.slp.useIPV6**. This parameter specifies whether IPv6 should be used for SLP.

  The default value is true.

In the following, the beginning of the slp.conf file is displayed:

```
#######################################################################
#
# OpenSLP configuration file
#
# Format and contents conform to specification in IETF RFC 2614 so the
# comments use the language of the RFC.  In OpenSLP, SLPD operates as an SA
# and a DA.  The SLP UA functionality is encapsulated by SLPLIB.
#
########################################################################

#----------------------------------------------------------------------
# Static Scope and Static DA Configuration
#----------------------------------------------------------------------

# This option is a comma delimited list of strings indicating the only
scopes
# a UA or SA is allowed when making requests or registering or the scopes a
# DA must support. (default value is "DEFAULT")
;net.slp.useScopes = myScope1, myScope2, myScope3

# Allows administrator to force UA and SA agents to use specific DAs.  If
# this setting is not used dynamic DA discovery will be used to determine
# which DAs to use.  (Default is to use dynamic DA discovery)
;net.slp.DAAddresses = myDa1,myDa2,myDa3

#----------------------------------------------------------------------
# DA Specific Configuration
#----------------------------------------------------------------------

# Enables slpd to function as a DA. Only a very few DAs should exist.  It
# is suggested that the administrator read the OpenSLP users guide before
# enabling this setting. Default is false.  Uncomment the line below to
# enable DA operation.
;net.slp.isDA = true

# A 32 bit integer giving the number of seconds for the DA heartbeat.
# Default is 3 hours (10800 seconds).  This property corresponds to
# the protocol specification parameter CONFIG_DA_BEAT [7].  Ignored
# if isDA is false.
;net.slp.DAHeartBeat = 10800
```

OpenSLP does not support all settings specified by RFC 2614.
The following options are not supported:

net.slp.serializedRegURL
net.slp.multicastTimeouts
net.slp.DADiscoveryTimeouts
net.slp.datagramTimeouts

The OpenSLP Registration Files

On SUSE Linux Enterprise Server 10, many applications already have integrated OpenSLP support.

With the SLP API (application programming interface), developers can easily add OpenSLP-based features to their programs.

The major function calls and more information can be found on RFC 2614.

Services are configured in registration files to make them available via OpenSLP.

The following ways of registration are possible:

- Static Registration via Files in /etc/slp.reg.d/
- Static Registration via /etc/slp.reg
- Dynamic Registration Using slptool

Static Registration via Files in /etc/slp.reg.d/

The best way to register a service for OpenSLP is to create a registration file for each service in the /etc/slp.reg.d/ directory.

The OpenSLP daemon slpd reads the registration files on startup, maintains all the registrations, and re-reads the files whenever the SIGHUP signal is received.

In order to be registered at the running Service Agent, the service needs to be running and have a valid registration file. Then, the Service Agent registers the service and can check the active state of the service when the lifetime expires.

Service Agent registration information is kept in memory and can be sent to Directory Agents.

To understand these registration files, you need to know the following:

- The OpenSLP Registration File Syntax
- The OpenSLP Service URL Syntax
- The OpenSLP Service Type Syntax

The OpenSLP Registration File Syntax

The registration file format is easy to understand.

Each registration consists of several lines with the format

#comment
;*comment*
service-url,language-tag,lifetime,service-type
scopes=*scope-list*
attrid=val1
attrid=val1,val2,val3

The options mean

- **service-url**. (required) This option defines the service URL.

  The syntax is described in "The OpenSLP Service URL Syntax" on 5-21.

- **language-tag**. (required) This option uses the (2-character) language tags as specified by RFC 1766 (such as en, fr, and de).

- **lifetime**. (required) This option defines the lifetime of the registration in seconds.

  The value must be between 0 and 65535.

  Use 65535 if you want the registration maintained for the life of slpd.

- **service-type**. (optional) This option defines the type of service being registered.

  This option is ignored by OpenSLP, because service-url must conform to the SLP Service URL format.

- **scope-list**. (optional) This option is a list of comma-delimited scopes to register the service in.

  If it is omitted, the service is registered in all scopes specified by the slp.conf file.

  By default, this is the DEFAULT scope on SLP version 2 systems.

- **attrid**. (optional) This option lists the attributes to register along with the service.

  Any string but scopes or SCOPES can be used as an attribute.

The OpenSLP Service URL Syntax

The service URL syntax looks like

service:*service-type*://*addrspec*

The options mean

- **service-type.** This option defines the service type as explained in the following paragraph.
- **addrspec**. This option can be just about anything you want that fits URL syntax (see RFC 2396) and can be translated as a network location.

The service: and :// strings are required.

If you want to use service URLs extensively, read RFC 2609.

The OpenSLP Service Type Syntax

The OpenSLP service type syntax is

abstract-type.naming-authority:concrete-type

The options mean

- **abstract-type**. This option describes the type of service.

  It should be a simple and short description.
- **naming-authority**. This option describes the name of the organization that named the service.

  It should be a unique name.

  The naming-authority is optional, but if it is omitted, IANA (Internet Assigned Numbers Authority) is assumed to be the naming authority, and IANA requires service types to be registered (see RFC 2609).

- **concrete-type**. A concrete-type is a kind of sub-type of the abstract type.

  For example, "printer" is an abstract type (owned by IANA) and "printer:lpr" is a concrete type (owned by IANA).

  This option is also optional.

The official definition of service type strings can be found in RFC 2609, "Service Templates and Service Schemes." If you want to work with well-known (IANA) service types, read this RFC.

RFC 2614 specifies a syntax for a registration file that is read by the OpenSLP daemon.

The following is the registration file for Samba (/etc/slp.reg.d/samba.reg):

```
#########################################################
#
# OpenSLP registration file
#
# register Samba and SWAT
#
#########################################################

# Register the samba server, if it is running
service:smb://$HOSTNAME,en,65535
tcp-port=139
description=Samba file and print server

# Register the web administration front-end for samba
service:Samba-Swat:http://$HOSTNAME:901,en,65535
tcp-port=901
description=Samba web administration front end
```

The lines are explained below:

- **service:smb://$HOSTNAME,en,65535**. This line describes the service URL:
  - **smb**. The abstract type.
  - **$HOSTNAME**. The hostname of the machine that offers the service.
  - **en**. The language (English).
  - **65535**. The service is registered the whole time while slpd is running.
- **tcp-port=139**. The service is provided on port 139.
- **description=Samba file and print server**. The service description.
- **service:Samba-Swat:http://$HOSTNAME:901,en,65535**. This line describes the service URL:
  - **Samba-Swat**. The abstract type.
  - **http**. The concrete type.
  - **$HOSTNAME**. The hostname of the machine that offers the service.
  - **901**. The service port.
  - **en**. The language (English).
  - **65535**. The service is registered the whole time while slpd is running.
- **tcp-port=901**. The service is provided on port 901.
- **description=Samba web administration front end**. The service description.

The following is the registration file for SSH (/etc/slp.reg.d/ssh.reg):

```
##########################################################
#
#
# OpenSLP registration file
#
# register SSH daemon
#
##########################################################

# Register the usual sshd, if it is running
service:ssh://$HOSTNAME:22,en,65535
tcp-port=22
description=Secure Shell Daemon

# ssh can get used to copy files with konqueror
# using the fish:/ protocol
service:fish://$HOSTNAME:22,en,65535
tcp-port=22
description=KDE file transfer via SSH
```

The lines are explained below:

- **service:ssh://$HOSTNAME:22,en,65535**. This line describes the service URL:
  - **ssh**. The abstract type.
  - **$HOSTNAME**. The hostname of the machine that offers the service.
  - **22**. The service port.
  - **en**. The language (English).
  - **65535**. The service is registered the whole time while slpd is running.
- **tcp-port=22**. The service is provided on port 22.
- **description=Secure Shell Daemon**. The service description.
- **service:fish://$HOSTNAME:22,en,65535**. This line describes the service URL:
  - **fish**. The abstract type.

- $HOSTNAME. The hostname of the machine that offers the service.
- **22**. The service port.
- **en**. The language (English).
- **65535**. The service is registered the whole time while slpd is running.

- **tcp-port=22**. The service is provided on port 22.
- **description=KDE file transfer via SSH**. The service description.

Static Registration via /etc/slp.reg

The file /etc/slp.reg can be used to register services which Service Agents that do not use the OpenSLP APIs.

Other than the services configured in /etc/slp.reg.d/, services not using the OpenSLP APIs cannot register themselves.

The file syntax is the same as in the /etc/slp.reg.d/* files.

You can find examples in the /etc/slp.reg file:

```
######################################################################
# OpenSLP registration file
#
# May be used to register services for legacy applications that do not use
# the SLPAPIs to register for themselves
#
# Format and contents conform to specification in IETF RFC 2614 so the
# comments use the language of the RFC.  In OpenSLP, SLPD operates as an SA
# and a DA.  The SLP UA functionality is encapsulated by the libslp
library.
#######################################################################
#comment
;comment
#service-url,language-tag,lifetime,[service-type]<newline>
#["scopes="scope-list<newline>]
#[attrid"="val1<newline>]
#[attrid"="val1,val2,val3<newline>]
#<newline>
#
# The following are examples entries for this file
#
##Register a OpenSLP testing service
#service:test.openslp://192.168.100.1,en,65535
#scopes=test1,test2
#description=OpenSLP Testing Service
#authors=mpeterson,jcarey

##Register ssh service
#service:ssh.openslp://192.168.100.1,en,65535
#use default scopes
#description="Secure Shell"

##Register telnet service with no attributes
#service:telnet.myorg://192.168.100.1,en,65535
#use default scopes

##Register vnc kdm service, can be used via krdc
#service:remotedesktop.kde:vnc://192.168.100.1:1,en,65535
#attrid=(type=shared),(username=joe),(fullname=Joe
User),(serviceid=1235456)
#description=KDE remote loginQ
```


Dynamic Registration Using slptool

If a service needs to be registered for OpenSLP from proprietary scripts, use the slptool command line frontend.

slptool provides two basic functionalities:

- It can be used to dynamically register services only for the runtime of slpd.
- It provides User Agent functionality to request services.

To list all available options and functions of slptool, enter

slptool - - help

or

slptool

The following is displayed:

```
Usage: slptool [options] command-and-arguments
   options may be:
      -v (or --version) displays version of slptool and OpenSLP
      -s (or --scope) followed by a comma separated list of scopes
      -l (or --language) followed by a language tag
   command-and-arguments may be:
      findsrvs service-type [filter]
      findsrvsusingiflist interface-list service-type [filter]
      unicastfindsrvs ip-address service-type [filter]
      findattrs url [attrids]
      unicastfindattrs ip-address url [attrids]
      findattrsusingiflist interface-list url [attrids]
      findsrvtypes [authority]
      unicastfindsrvtypes [authority]
      findsrvtypesusingiflist interface-list [authority]
      findscopes
      register url [attrs]
      deregister url
      getproperty propertyname
Examples:
   slptool register service:myserv.x://myhost.com
"(attr1=val1),(attr2=val2)"
   slptool findsrvs service:myserv.x
   slptool findsrvs service:myserv.x "(attr1=val1)"
   slptool findsrvsusingiflist 10.77.13.240,192.168.250.240
service:myserv.x
   slptool findsrvsusingiflist 10.77.13.243 service:myserv.x
"(attr1=val1)"
   slptool unicastfindsrvs 10.77.13.237 service:myserv.x
   slptool unicastfindsrvs 10.77.13.237 service:myserv.x "(attr1=val1)"
   slptool findattrs service:myserv.x://myhost.com

...
```

Objective 4 Use SLP Frontends

slptool is a simple command line program that can be used

- To introduce SLP services to the Service Agents dynamically (only valid during runtime).

  or

- To discover SLP services on the network.

The first functionality has already been discussed.

Here is an example how to use the second functionality:

To find out which server provides SSH services, do the following:

1. On the command line, enter

   slptool findsrvtypes

   A list of services similar to the following is displayed:

```
service:ntp
service:kde.sld.suse
service:smtp
service:ssh
service:fish
service:remotedesktop.kde:vnc
service:remotedesktop.java:http
service:ldap
service:smb
service:install.suse:nfs
service:install.suse:ftp
service:install.suse:http
service:smdr.novell
service:bindery.novell
service:ndap.novell
service:wbem:https
```

2. Use **service:ssh** to get information on the servers that provide the SSH services. To do this, enter

   slptool findsrvs service:ssh

The list of hosts that provide the SSH service is displayed:

```
service:ssh://da1.digitalairlines.com:22,65535
service:ssh://da3.digitalairlines.com:22,65535
service:ssh://da4.digitalairlines.com:22,65535
service:ssh://da10.digitalairlines.com:22,65535
service:ssh://da11.digitalairlines.com:22,65535
```

Enter **slptool --help** to get a list of all available options and functions.

As described above (in “Dynamic Registration Using slptool” on 5-27), slptool can also be called from scripts that process SLP information.

Exercise 5-1 Provide the daytime Service with OpenSLP

In this exercise, you learn how to configure SLP on your server with a simple service independent of the network wide SLP design considerations.

You will find this exercise in the workbook.

(End of Exercise)

Summary

Objective	Summary
1. Understand what OpenSLP Is	Service Location Protocol (SLP) provides the dynamic discovery of services. On SUSE Linux Enterprise Server 10, OpenSLP is available. It's an Open Source implementation of SLP v2. OpenSLP registers service information in a database and allows SLP clients to query the database to find these services.
2. Understand the SLP Agents	OpenSLP uses agents to represent applications and services on the network: ■ User Agent ■ Service Agent ■ Directory Agent The dialog between these agents is limited to very simple exchanges of request and reply messages. SLP is a unicast and multicast protocol. The messages can be sent ■ To one agent at a time (unicast) or ■ To all agents (that are listening) at the same time (multicast)

Objective	Summary
3. Configure OpenSLP	To configure OpenSLP, you need to know the following: ■ The OpenSLP Daemon slpd The OpenSLP daemon is /usr/sbin/slpd. On SUSE Linux Enterprise Server 10, the daemon is installed and active by default. Its init script is /etc/init.d/slpd. To stop the daemon, enter **/etc/init.d/slpd stop** or **rcslpd stop** To start the daemon, enter **/etc/init.d/slpd start** or **rcslpd start** To restart the daemon, enter **/etc/init.d/slpd restart** or **rcslpd restart** To check the daemon's status, enter **/etc/init.d/slpd status** or **rcslpd status**

Objective	**Summary**
3. Configure OpenSLP (*continued*)	■ The OpenSLP Configuration File The OpenSLP configuration file is /etc/slp.conf. The daemon slpd reads this file when it is started. The file contains parameters in the format ***parameter=value*** ■ The OpenSLP Registration Files Services are registered in registration files to make them available via OpenSLP. The possible ways of registration are: ■ Static Registration via Files in /etc/slp.reg.d/ ■ Static Registration via /etc/slp.reg ■ Dynamic Registration Using slptool
4. Use SLP Frontends	**slptool** is a simple command line program that can be used to ■ Introduce SLP services to the Service Agents dynamically (only valid during runtime) or ■ Discover SLP services on the network

SECTION 6 Monitor Network Traffic

After deploying your server in a network, it's important to monitor your server network communications.

If you deploy your server in an existing network where you already have already a network-wide monitoring and management system based on SNMP, we recommend adding the new server to that framework by configuring the package net-snmp (based on the CMU 2.1.2.1 cmu-snmp code).

This type of configuration must follow the policies you established in your network-wide SNMP configuration.

However, if you do not have an SNMP-based monitoring system, this section introduces you to some useful tools for monitoring network traffic on your SUSE Linux Enterprise Server 10.

While these tools do not provide a management functionality like SNMP, they do provide capabilities critical for troubleshooting potential networking issues.

Objectives

1. Use ntop to Monitor Network Traffic
2. Use Nagios to Monitor Hosts, Services, and Network
3. Troubleshoot the Network with Ethereal and tcpdump

Objective 1 Use ntop to Monitor Network Traffic

ntop is a web-based traffic monitor that shows network usage, similar to what the top command does. It is not included on SUSE Linux Enterprise Server 10 anymore. But it is included on the ***Course*** DVD.

In this objective you will learn the following:

- Configure ntop
- Use ntop

Configure ntop

To install ntop, select the ntop package in the software installation of YaST.

The configuration file of ntop is /etc/sysconfig/ntop. The available options are:

- **NTOPD_IFACE.** Specifies the network interface used by ntop. If you enter more than one interface, use commas to separate the entries.
- **NTOPD_PORT.** Specifies the port used by ntop (default: 3000).
- **NTOPD_SSL_PORT.** Define the SSL port. You have to generate a certificate to run ntop with this option.
- **NTOP_USER.** Define the user to run ntop. This should not be root!
- **NTOP_ARGS.** Additional arguments when starting ntop with the init script /etc/init.d/ntop or rcntop.

See **man 8 ntop** for all available command line options.

Use ntop

Before starting ntop, you have to set the administrator password:

```
da51:~ # ntop -A -u wwwrun
Tue May 24 04:32:57 2005  IniInitializing gdbm databases
Tue May 24 04:32:57 2005  Now running as requested user 'wwwrun' (30:8)

Please enter the password for the admin user:
Please enter the password again:
Tue May 24 04:32:59 2005  Admin user password has been set
da51:~ #
```

The -A parameter starts ntop, sets the admin password, and quits ntop. -A will prompt the user for the password.

The -u parameter specifies the user ntop should run as after it initializes.

The password is stored in the file /var /lib/ntop/ntop_pw.db.

ntop must normally be started as root so that it has sufficient privileges to open the network interfaces in promiscuous mode and to receive raw frames.

After starting ntop, the processes run with the UID of the user you specify on the command line. On SUSE Linux Enterprise Server 10, normally the user wwwrun is used for running ntop.

This section covers the following:

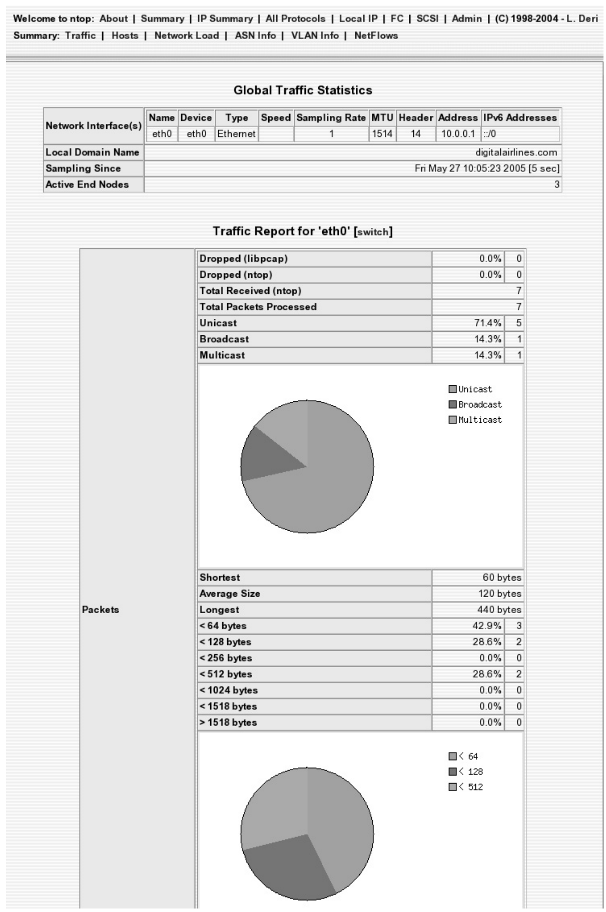
- Use ntop in Web Mode
- Command Options of ntop

Use ntop in Web Mode

After entering **ntop**, you can access the ntop web interface by entering the following URL in a web browser:

http://hostname:3000

Figure 6-1



In the top line, there are eight sections:

- **About**. This section contains
  - An overview of the ntop configuration.
  - Help.
- **Summary.** This section contains
  - A summary about network traffic.
  - Host network information.
  - Information on the network load.
  - Autonomous systems traffic statistics.
  - VLAN traffic statistics.
  - Network flows. (A flow is a stream of captured packets that match a specified rule.)
- **IP Summary.** This section contains
  - Information on network traffic of each host.
  - Multicast statistics.
  - Statistics for all domains.
  - Distribution of the IP protocols.
  - Local and remote IP traffic.
- **All Protocols.** This section contains
  - Protocols used by each host.
  - Network throughput.
  - Timetable with the network activity of the hosts.
- **Local IP.** This section contains
  - Information about local subnet routers.
  - Local protocol usage.
  - Active TCP sessions.
  - Fingerprints of the local hosts.
  - Network services provided by the local hosts.

    - IP subnet traffic matrix.

- **FC** (Fibre Channel). This section contains
    - Information on traffic.
    - Information on throughput.
    - Information on activity.
    - Information on hosts.
    - Traffic per port.
    - Sessions.
    - VSAN traffic.
- **SCSI.** This section contains
    - Sessions.
    - Latencies.
    - Status information.
    - Task management information.
- **Admin.** This section contains
    - Plugins.
    - Dump.
    - Log files.
    - Configuration.

Command Options of ntop

If you open the man page of ntop (man 8 ntop), you can see that there are numerous options.

The most important options are

- **-r *delay***. Specifies the delay (in seconds) between automatic screen updates for those generated HTML pages that support them.

- **-f** ***file***. By default, ntop captures traffic from network interface cards (NICs) or from netFlow/sFlow probes. If you specify -f, ntop will not capture any traffic from NICs and reads the content of the file instead.

  To capture network traffic, use **tcpdump** or **Ethereal** as described later in this section.

- **-n**. This parameter causes ntop to skip DNS resolution, showing only numeric IP addresses instead of the symbolic names.

- **-p** ***protocols***. This parameter specifies the TCP/UDP protocols that ntop will monitor.

  The format is ***label=protocol list***[**,** ***label=protocol list***], where ***label*** is used to symbolically identify the ***protocol list***.

  The format of ***protocol list*** is ***protocol***[**|protocol**], where ***protocol*** is either a valid protocol specified inside the /etc/services file or a numeric port range (such as 80 or 6000–6500).

  Example: **-p FTP=ftp|ftp-data**

- **-i** ***interface***. Specifies the network interfaces to be used by ntop for network monitoring.

- **-w** ***port***. The ntop web server is embedded in the application. These parameters specify the port (and optionally the address) of the ntop web server.

  The default port is 3000. The option **-W** operates similarly, but controls the port for the HTTPS connections.

- **-d**. This parameter causes ntop to become a daemon in the background.

- **-m** ***subnets***. This parameter allows the user to define additional networks and subnetworks whose traffic is also considered local in ntop reports.

- **-l**. This parameter causes a dump file to be created of the network traffic captured by ntop in tcpdump (pcap) format.

Exercise 6-1 Use ntop

In this exercise, you install and start ntop.

You will find this exercise in the workbook.

(End of Exercise)

Objective 2 Use Nagios to Monitor Hosts, Services, and Network

Nagios is a program for monitoring hosts and services. Nagios can be used to:

- Monitor host-based activity (such as use of disk space and running processes).
- Monitor network services (such as SMTP, POP3, and HTTP).
- Notify a person in case problems appear and disappear (using email or paging).
- Provide a plugin interface for user-developed monitoring methods.
- Provide access to the status via a webbrowser.

Several packages can be used with Nagios. We will use the following packages in this chapter:

- **nagios**. The base package of Nagios.
- **nagios-nsca**. The Nagios service check acceptor.
- **nagios-plugins**. Plugins needed for Nagios checks.
- **nagios-plugins-extra**. More plugins for Nagios checks.
- **nagios-www**. A web server for Nagios.

The following topics are covered in this section:

- Directories Used by Nagios
- Basic Configuration of Nagios
- Configuration of Contacts and Contact Groups
- Configure the Hosts and Host Groups
- Configure the Services
- Configure Other Resources

- Configure the Nagios Web Server
- Use Nagios

Directories Used by Nagios

Nagios uses the following directories by default:

- **/etc/nagios/**. Configuration files
- **/usr/lib/nagios/plugins/**. Plugins for checking hosts and services
- **/usr/lib/nagios/cgi/**. CGI scripts for the web server
- **/var/log/nagios/**. Log files of Nagios
- **/var/spool/nagios/**. Spool file of Nagios
- **/etc/apache2/conf.d/** Configuration files of the web server
- **/usr/share/nagios/** Content files of the web server

Basic Configuration of Nagios

All configuration files for Nagios are located in the directory /etc/nagios/. The main configuration file is /etc/nagios/nagios.cfg. This file can include additional configuration files.

The file /etc/nagios/nagios.cfg contains a lot of comments describing the different configuration options. Most of the options can be used as provided; we will describe only the most important options.

The definition of the objects to monitor is done by including configuration files using the **cfg_file** statement. We will discuss those statements in the respective sections.

- **log_file=/var/log/nagios/nagios.log**. Location of the logfile for Nagios. Service and host events are logged to this file. This should be the first option specified in the configuration file.
- **cfg_file=/etc/nagios/checkcommands.cfg**. This statement reads the definition of the plugin commands (service and host check commands).
- **cfg_file=/etc/nagios/misccommands.cfg**. This statement reads the definition of miscellaneous commands (notification and event handler commands, etc.)
- **status_file=/var/log/nagios/status.log**. This is where the current status of all monitored services and hosts is stored. The contents of the status file are read and processed by the CGI scripts. The contents are deleted every time Nagios restarts.
- **nagios_user=daemon** and **nagios_group=daemon**. These two statements define the effective user and group ID to be used for the Nagios processes. You may specify a user name/group name or a UID/GID.
- **check_external_commands=1**. If you want to use the Nagios web server, you need to run the CGI scripts provided with the package. To do so, you have to set this option to **1**. The default value is **0**, disabling this functionality.
- **command_file=/var/spool/nagios/nagios.cmd**. This file will be used to look for external commands. In case of the CGI scripts, this is a named pipe where the commands will be written to.
- **log_rotation_method=d**. Rotation method for the log file (/var/log/nagios/nagios.log). By default, the logfile will be rotated daily. Values can be:
    - **n**. None – don't rotate the log
    - **h**. Hourly rotation (top of the hour)
    - **d**. Daily rotation (midnight every day)
    - **w**. Weekly rotation (midnight on Saturday evening)

- **m**. Monthly rotation (midnight last day of month)

  It may be useful to deactivate the log file rotation here. The file /etc/logrotate.d/nagios is checked on a daily basis to monitor the log file. If the log file grows to more than 1024 KB, it will be rotated.

- **log_archive_path=/var/log/nagios/archives**. If rotating logfiles is activated, the logfiles will be moved to this directory. If you set **log_rotation_method=n**, this directive will have no effect.

- **use_syslog=1**. If you want messages logged via the syslog daemon as well, set this option to **1**. If not, set it to **0**.

Configuration of Contacts and Contact Groups

A contact definition is used to identify someone who should be contacted in the event of a problem on your network. These contacts have to be assigned to a contact group (see below) which is informed by Nagios about certain problems. All contacts should be defined in a special file, which is included with the statement

```
cfg_file=/etc/nagios/contacts.cfg
```

An example for a contact definition is shown here:

```
# contact definition for kbailey
define contact{
        contact_name                    kbailey
        alias                           Kate Bailey
        service_notification_period     workhours
        host_notification_period        workhours
        service_notification_options    c,r
        host_notification_options       d,r
        service_notification_commands   notify-by-email
        host_notification_commands      host-notify-by-email
        email                          kbailey@digitalairlines.com
}
```

The parameters for defining a contact are as follows:

- **contact_name**. Used to define a short name to identify the contact. It is referenced in contact group definitions (see below).
- **alias**. Used to define a longer name or description for the contact.
- **service_notification_period**. Used to specify the short name of the time period during which the contact can be notified about service problems or recoveries. You can think of this as an "on call" time for service notifications for the contact.
- **host_notification_period**. Used to specify the short name of the time period during which the contact can be notified about host problems or recoveries. You can think of this as an "on call" time for host notifications for the contact.
- **service_notification_options**. Used to define the service states for which notifications can be sent out to this contact. Valid options are a combination of one or more of the following:
  - **w**. Notify on WARNING service states.
  - **u**. Notify on UNKNOWN service states.
  - **c**. Notify on CRITICAL service states.
  - **r**. Notify on service recoveries (OK states).

  If you specify **n** (none) as an option, the contact will not receive any type of service notifications.
- **host_notification_options**. Used to define the host states for which notifications can be sent out to this contact. Valid options are a combination of one or more of the following:
  - **d**. Notify on DOWN host states.
  - **u**. Notify on UNREACHABLE host states.
  - **r**. Notify on host recoveries (UP states).

  If you specify **n** (none) as an option, the contact will not receive any type of host notifications.

- **host_notification_commands**. Used to define a list of the short names of the commands used to notify the contact of a host problem or recovery. Multiple notification commands should be separated by commas.
- **email**. Used to define an email address for the contact. Depending on how you configure your notification commands, it can be used to send out an email alert to the contact. This email address must exist on the system (/etc/aliases).

All contacts must be members of at least one contact group. The definition of the contact groups is included with the statement

```
cfg_file=/etc/nagios/contactgroups.cfg
```

An example of a definition of a contact group looks like this:

```
# 'admins' contact group definition
define contactgroup{
        contactgroup_name       admins
        alias                   Administrators
        members                 kbailey,jgoldman
}
```

The parameters for defining a contact group are as follows:

- **contactgroup_name**. A short name used to identify the contact group.
- **alias**. A longer name or description used to identify the contact group.
- **members**. A list of the short names of contacts that should be included in this group. Multiple contact names should be separated by commas.

Configure the Hosts and Host Groups

The hosts you want to monitor with Nagios are defined in the file /etc/nagios/hosts.cfg. A host definition is used to define a physical machine: a server, a workstation, a network device (such as routers or switches), etc. The list of host definitions is included using the statement

```
cfg_file=/etc/nagios/hosts.cfg
```

At the beginning is the definition of a host template:

```
# Generic host definition template
define host{
        name                            generic-host
        notifications_enabled           1
        event_handler_enabled           1
        flap_detection_enabled          1
        process_perf_data               1
        retain_status_information       1
        retain_nonstatus_information    1
        register                        0
}
```

This template is referenced in other host definitions and allows you to use definitions which are already specified.

We do not go into the details here; this template is just activating different actions. The last line (register 0) is needed to indicate that this is not a real host but just a template.

An example for a host definition is shown here:

```
# 'da10' host definition
define host{
        use                     generic-host
        host_name               da10
        alias                   SLES 10 #1
        address                 10.0.0.10
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   120
        notification_period     24x7
        notification_options    d,u,r
}
```

The parameters for the host definition are as follows:

- **use**. Name of host template to use.
- **host_name**. Used to define a short name that is used to identify the host. It is used in host group and service definitions to reference this particular host. Hosts can have multiple services (which are monitored) associated with them.
- **alias**. Used to define a longer name or description that is used to identify the host. It is provided to allow you to more easily identify a particular host.
- **address**. Used to define the address of the host. Normally, this is an IP address, although it could really be anything you want (so long as it can be used to check the status of the host). You can use an FQDN to identify the host instead of an IP address, but if DNS services are not available, this could cause problems.

- **check_command**. Used to specify the short name of the command that should be used to check if the host is up or down. Typically, this command would try to ping the host to see if it is alive. The command must return a status of OK (0) or Nagios will assume the host is down. If you leave this argument blank, the host will not be checked—Nagios will always assume the host is up. This is useful if you are monitoring printers or other devices that are frequently turned off. The maximum amount of time that the notification command can run is controlled by the host_check_timeout option.
- **max_check_attempts**. Used to define the number of times that Nagios will retry the host check command if it returns any state other than an OK state. Setting this value to 1 will cause Nagios to generate an alert without retrying the host check again. Note: If you do not want to check the status of the host, you must still set this to a minimum value of 1. To bypass the host check, just leave the check_command option blank.
- **notification_interval**. Used to define the number of time units to wait before re-notifying a contact that this server is still down or unreachable. Unless you've changed the interval_length directive from the default value of 60, this number will mean minutes. If you set this value to 0, Nagios will not re-notify contacts about problems for this host—only one problem notification will be sent out.
- **notification_period**. Used to specify the short name of the time period during which notifications of events for this host can be sent out to contacts. If a host goes down, becomes unreachable, or recovers during a time which is not covered by the time period, no notifications will be sent out.
- **notification_options**. Used to determine when notifications for the host should be sent out. Valid options are a combination of one or more of the following:
  - **d**. Send notifications on a DOWN state.
  - **u**. Send notifications on an UNREACHABLE state.

- **r**. Send notifications on recoveries (OK state).

  If you specify **n** (none) as an option, no host notifications will be sent out.

  Example: If you specify **d,r** in this field, notifications will only be sent out when the host goes DOWN and when it recovers from a DOWN state.

All hosts must be a member of at least one host group. The definition of host groups is included using the statement

```
cfg_file=/etc/nagios/hostgroups.cfg
```

An example for a definition of a host group looks like this:

```
# 'linux-boxes' host group definition
define hostgroup{
        hostgroup_name  linux-boxes
        alias           Linux Servers
        contact_groups  admins
        members         da10,da11
}
```

The parameters for defining a host group are as follows:

- **hostgroup_name**. Used to define a short name that is used to identify the host group.
- **alias**. Used to define a longer name or description that is used to identify the host group. It is provided in order to allow you to more easily identify a particular host group.
- **contact_groups**. A list of the short names of the contact groups that should be notified whenever there are problems (or recoveries) with any of the hosts in this host group. Multiple contact groups should be separated by commas.
- **members**. A list of the short names of hosts that should be included in this group. Multiple host names are separated by commas.

Configure the Services

The services to check are defined in a separate file. This file is included with the statement

```
cfg_file=/etc/nagios/services.cfg
```

You have to specify each service on each host as a separate entry in this file.

At the beginning is the definition of a service template:

```
# Generic service definition template
define service{
        name                            generic-service
        active_checks_enabled           1
        passive_checks_enabled          1
        parallelize_check               1
        obsess_over_service             1
        check_freshness                 0
        notifications_enabled           1
        event_handler_enabled           1
        flap_detection_enabled          1
        process_perf_data               1
        retain_status_information       1
        register                        0
}
```

This template is referenced in other service definitions and allows you to use definitions which are already specified.

Again, we do not go into the details of this template. The last line (register 0) indicates that this is not a real service, just a template.

An example of the definition for a network service is shown here:

```
# Service definition
define service{
        use                             generic-service
        host_name                       da10
        service_description             SMTP
        is_volatile                     0
        check_period                    24x7
        max_check_attempts              3
        normal_check_interval           3
        retry_check_interval            1
        contact_groups                  admins
        notification_interval           120
        notification_period             24x7
        notification_options            w,u,c,r
        check_command                   check_smtp
}
```

An example of the definition for a local host-based service is shown here:

```
# Service definition
define service{
        use                             generic-service
        host_name                       da10
        service_description             Check local disk usage
        is_volatile                     0
        check_period                    24x7
        max_check_attempts              3
        normal_check_interval           5
        retry_check_interval            1
        contact_groups                  admins
        notification_interval           120
        notification_period             24x7
        notification_options            c,r
        check_command                   check_local_disk!80%!90%!/
        }
```

The parameters used are as follows:

- **use**. Name of the service template to use.

- **host_name**. Reference to the host name in /etc/nagios/hosts.cfg.
- **service_description**. Description of the check to perform.
- **check_period**. Period during which this service should be checked. The name provided here refers to the definition in /etc/nagios/timeperiods.cfg.
- **max_check_attempts**. The maximum number of check attempts which should be done.
- **normal_check_interval**. The interval length at which the service should be checked.
- **retry_check_interval**. In case the check could not be performed successfully, specify the time after which the next attempt should be made.
- **contact_groups**. The contact group to be notified.
- **notification_interval**: Used to define the number of time units to wait before re-notifying a contact that this server is still down or unreachable. Unless you've changed the interval_length directive from the default value of 60, this number will mean minutes. If you set this value to 0, Nagios will not re-notify contacts about problems for this host—only one problem notification will be sent out.
- **notification_options**. Definition of the service states when a notification should be sent out:
  - **w**. Notify on WARNING service states.
  - **u**. Notify on UNKNOWN service states.
  - **c**. Notify on CRITICAL service states.
  - **r**. Notify on service recoveries (OK states).
- **check_command**. The command to be used for the check. The structure of the command is defined in the file /etc/nagios/checkcommands.cfg (see below). Depending on this structure, different parameters have to be supplied to the command. The parameters are separated by the exclamation mark.

Here is the default definition of the check_local_disk command:

```
# 'check_local_disk' command definition
define command{
        command_name    check_local_disk
        command_line    $USER1$/check_disk -w $ARG1$ -c $ARG2$ -p $ARG3$
}
```

The parameters for the definition are:

- **command_name**. The name of the command used in the service definitions.
- **command_line**. The plugin that is started with the required parameters. In this example definition, three parameters are required to use this plugin correctly.

The meaning of the parameters can be determined by calling the plugin with the option **--help**:

```
da51:~# /usr/lib/nagios/plugins/check_disk --help
...
This plugin will check the percent of used disk space on a mounted
file system and generate an alert if percentage is above one of the
threshold values.

Usage: check_disk -w limit -c limit [-p path | -x device] [-t timeout]
[-m] [-e]
+[--verbose]
       check_disk (-h|--help)
       check_disk (-V|--version)
```

In the service definition above, the command is called as

check_local_disk!80%!90%!/

This means a warning message will be created when the disk space usage exceeds 80% and a critical message will be generated when the disk space usage exceeds 90%. The file system to check is the root file system /.

Configure Other Resources

Two configuration files are used to define additional resources. The file which defines the names of the time periods to use is included with the statement:

```
cfg_file=/etc/nagios/timeperiods.cfg
```

The names of the time definitions are used in the services and the hosts definitions.

A configuration file for basic settings is included with the statement

```
resource_file=/etc/nagios/resource.cfg
```

The file is used for setting variables which are used in the other configuration files. By default, only one directive is activated:

```
# Sets $USER1$ to be the path to the plugins
$USER1$=/usr/lib/nagios/plugins
```

The variable $USER1$ is used in the configuration file for commands to do the checks (/etc/nagios/checkcommands.cfg).

Configure the Nagios Web Server

To use the web server of Nagios, some configuration has to be done. The web server is designed to use the Apache web server; on SUSE Linxu Enterprise Server 10, its Apache version 2. By default, access to the web server is only allowed from localhost. The configuration file is /etc/apache2/conf.d/nagios.conf:

```
ScriptAlias /nagios/cgi-bin/ /usr/lib/nagios/cgi/
<Directory /usr/lib/nagios/cgi/>
   Options ExecCGI
   order deny,allow
   deny from all
   allow from 127.0.0.1
</Directory>

Alias /nagios/ /usr/share/nagios/
<Directory /usr/share/nagios/>
   Options None
   order deny,allow
   deny from all
   allow from 127.0.0.1
</Directory>
```

To facilitate access to the CGI scripts (see below), it is advisable to require user authentication when accessing the Nagios web server. If security reasons requireyou to limit access to the web server to special IP addresses, this can be defined as well.

The easiest way to configure user authentication is to modify /etc/apache2/conf.d/nagios.conf in this way:

```
ScriptAlias /nagios/cgi-bin/ /usr/lib/nagios/cgi/
<Directory /usr/lib/nagios/cgi/>
   AllowOverride AuthConfig
   Options ExecCGI
   order deny,allow
   allow from all
</Directory>

Alias /nagios/ /usr/share/nagios/
<Directory /usr/share/nagios/>
   AllowOverride AuthConfig
   Options None
   order deny,allow
   allow from all
</Directory>
```

The line AllowOverride AuthConfig allows Nagios to use a file .htaccess in the directoy which should be protected. The contents of this file could look like this:

```
AuthName "Nagios Access"
AuthType Basic
AuthUserFile /etc/apache2/nagios
require valid-user
```

Only valid users which are defined in the file /etc/apache2/nagios are allowed to connect to the Nagios web server. You create this file with the command **htpasswd2**:

```
da51:~# htpasswd2 -c /etc/apache2/nagios nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
```

Adding more users to the file is done using **htpasswd2** without the option **-c**.

The directory /usr/lib/nagios/cgi/ contains a collection of CGI scripts which are used to access the Nagios web server and to display the status of the services and hosts monitored by Nagios. By default, access to these CGI scripts is denied for everybody. You have to allow access to the scripts by modifying the file /etc/nagios/cgi.cfg.

If you use the user name nagiosadmin for connecting to the web server, there is not much to modify in order to allow access to the CGI scripts.

By default, user authentication is required to access the CGI scripts:

```
use_authentication=1
```

All configuration lines defining authorized user names start with "authorized_for_". An example is

```
#authorized_for_system_information=nagiosadmin,theboss,jdoe
```

To activate the statement, just remove the comment character (#) in the first column and modify the list of allowed user names. This list is a comma-separated list of user names. By default, they all include the user name nagiosadmin.

Use Nagios

Nagios is started using the command

rcnagios start

In case of configuration errors, you will see a message pointing to the file /var/log/nagios/config.err, which contains a description of the error.

To access the Nagios web server, point your browser to the directory /nagios, e.g.

http://da51.digitalairlines.com/nagios/

Configurating of https to access Nagios is also possible but is not described here.

Exercise 6-2 Use Nagios

In this exercise, you install and configure Nagios.

You will find this exercise in the workbook.

(End of Exercise)

Objective 3 Troubleshoot the Network with Ethereal and tcpdump

- Use Ethereal
- Use tcpdump

Use Ethereal

Ethereal is a network protocol analyzer. It allows you to examine data from a live network or from a captured file on disk.

You can interactively browse the capture data, viewing summary and detail information for each packet.

Ethereal has a powerful display filter language and the ability to view a reconstructed stream of a TCP session.

To install Ethereal, select the ethereal package in the YaST software installation module; then enter **ethereal**.

To use the features of the program, you need to be logged in as root.

In this objective, the following is covered:

- Capture Traffic
- The Ethereal Frontend
- Filter Display

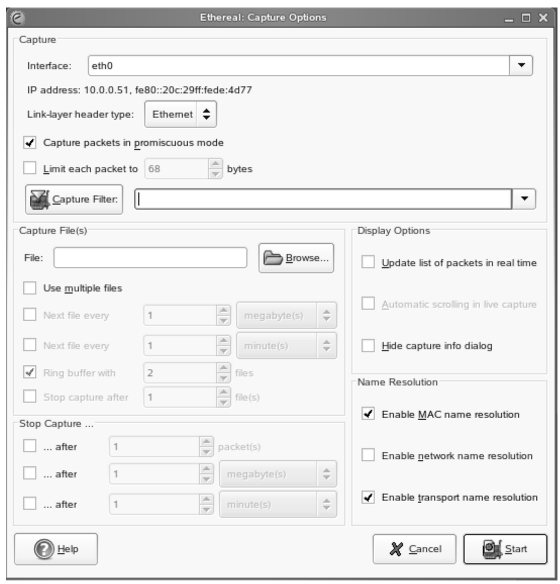
Capture Traffic

If you start Ethereal, the dialog shows only a menu bar and a tool bar at the top. To open a file containing a capture of network traffic, select
File > Open

To capture actual network traffic, select **Capture > Options**.

The following dialog appears:

Figure 6-2



From the **Interface** menu, you can select the interface (the traffic you want to capture).

In **Capture Filter**, you can enter a filter rule.

Possible commands for the filter rule are

- [**src**|**dst**] **host** ***host***. Allows you to filter a host IP address or name.

If you add **src** or **dst**, you can specify that you are only interested in source or destination addresses.

- **ether** [**src**|**dst**] **host** ***host***. Allows you to filter Ethernet host addresses.

  If you add **src** or **dst**, you can specify that you are only interested in source or destination addresses.

- **gateway hos**t ***host***. Allows you to filter packets that used the host as a gateway (where the Ethernet source or destination was host but neither the source nor destination IP address was host).

- [**src**|**dst**] **net** ***net*** [**mask <*mask*>**|**len** ***len***]. Allows you to filter network numbers.

  If you add **src** or **dst**, you can specify that you are only interested in source or destination addresses.

  In addition, you can specify either the netmask or the CIDR (Classless InterDomain Routing) prefix for the network if it is different from your own.

- [**tcp**|**udp**] [**src**|**dst**] **port** ***port***. Allows you to filter TCP and UDP port numbers.

  You can optionally procede this parameter with the keywords **src** or **dst** and **tcp** or **udp**. These parameters allow you to specify that you are only interested in source or destination ports and TCP or UDP packets respectively.

  The keywords **tcp** or **udp** must appear before **src** or **dst**.

- **less**|**greater** ***length***. Allows you to filter packets whose length was less than or equal to the specified length, or greater than or equal to the specified length, respectively.

- **ip**|**ether proto** ***protocol***. Allows you to filter the specified protocol at either the Ethernet layer or the IP layer.

- ***ether***|**ip broadcast**|**multicast**. Allows you to filter either Ethernet or IP broadcasts or multicasts.

- ***expr* relop *expr***. Allows you to create complex filter expressions that select bytes or ranges of bytes in packets.

You can combine these commands with **and**, **or**, or **not**. You can also use brackets.

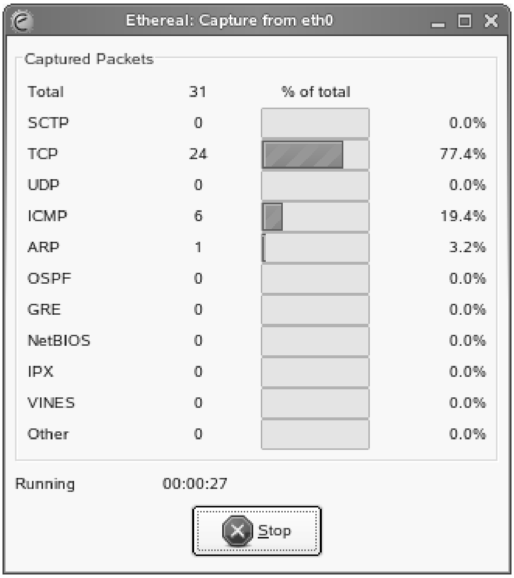
If you want to write the captured information into a file, use the options in the Capture File(s) frame. You can split the captured information into several files.

By default, you have to stop the capture process with a mouse click. In the Stop Capture frame, you can choose to stop the capturing after a number of packets has been captured, a level of used memory, or after a certain time.

To start capturing select **Start**.

During the capture process, the following dialog shows how many packets have been captured.

Figure 6-3



If you want to see the captured packets in real time, use the options of the Display Options frame.

The Name Resolution options (in the bottom right corner) are also useful. You can specify which names should be resolved.

After selecting **OK**, the capture begins.

To stop the capture process, select **Stop**.

The Ethereal Frontend

After capturing packages or opening a file with captured network traffic, the Ethereal dialog shows three frames:

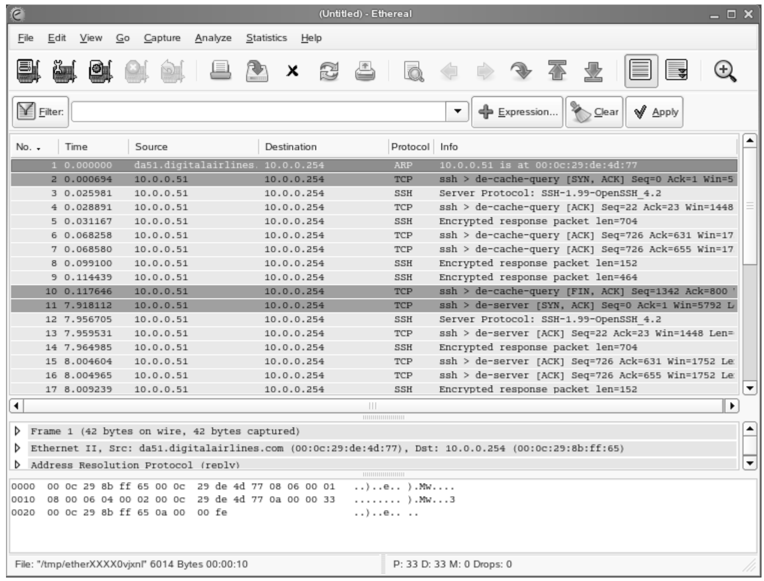
- **packet list** (top). This frame contains a table showing the following information:
  - **No.** Number of the packet.
  - **Time**. Time when the packet is received.
  - **Source**. Sender of the packet.
  - **Destination**. Receiver of the packet.
  - **Protocol**. Used protocol.
  - **Info**. More information about the content of the packet.
- **packet details** (middle). Shows the content of the packet.

  Select the triangle symbol to open a menu.

  The structure of menus depends on the kind of protocol.
- **packet bytes** (bottom) The bytes of the packet written in hexadecimal and ASCII characters.

Below the packet bytes frame is a line for configuring the view filter.

Figure 6-4



Filter Display

You can filter the displayed packets. You can enter a filter expression in the text box next to the **Filter** button in the top line.

The expressions for display filters are different from the capture filter expressions.

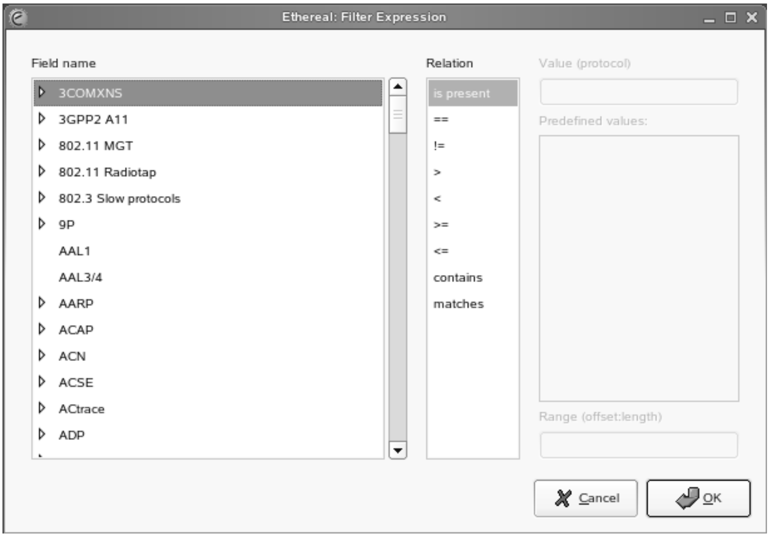
Possible expressions are:

- **eth.addr==08.00.08.15.ca.fe**. Display all traffic to and from the Ethernet address 08.00.08.15.ca.fe.
- **ip.addr==192.168.0.10**. Display all traffic to and from the IP address 192.168.0.10.

- **tcp.port==80**. Display all traffic to and from the TCP port 80 (HTTP) of all machines.
- **ip.addr==192.168.0.10 && tcp.port!=80**. Display all traffic to and from 192.168.0.10 except HTTP.

Ethereal provides a powerful tool to construct display filters. Select **Expressions** in the filter line.

Figure 6-5



In the left frame (labeled **Field name**), you can see a list of options you can filter for. These fields are grouped for protocols or services.

In the middle frame, you can specify the relationship. The relationship shown depend from the selected field.

Depending on the selected relationship, you can specify a value in the text box **Value** in the top right column. The dialog shows what kind of value is expected.

If you select **View > Coloring Rules**, you can create a display filter that colorizes the matches in a certain color.

When you want to see the traffic of a whole network session (e.g., a whole telnet session), it is uncomfortable having to select each package from the packet list. Instead, you can select a TCP packet in the session and choose **Analyze > Follow TCP Stream**.

Exercise 6-3 Use Ethereal

In this exercise, you use Ethereal and capture all traffic being sent to and from da2.

You will find this exercise in the workbook.

(End of Exercise)

Use tcpdump

This software can read all or certain packets going over the ethernet. tcpdump is often used to save the network traffic in a file.

tcpdump is installed during the default installation of SUSE Linux Enterprise Server 10.

The simplest way of using tcpdump is to enter just **tcpdump**.

To quit the capture, press **Ctrl + C**. A short summary is shown.

```
da51:~ # tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
16:52:50.488641 802.1d config 8000.00:80:2d:e2:9d:13.8013 root
8000.00:80:2d:e2:9d:13 pathcost 0 age 0 max 20 hello 2 fdelay 15
16:52:51.296790 (NOV-802.2) f4a33f8b.00:50:8b:d6:9b:8b.9001 >
00000000.ff:ff:ff:ff:ff:ff.9001: ipx-#9001 43

2 packets captured
2 packets received by filter
0 packets dropped by kernel
da51:~ #
```

The most important options of tcpdump are

- **-c *number***. Exit after receiving ***number*** of packets.
- **-F *file***. Use ***file*** as input for the filter expression.

  An additional expression given on the command line is ignored.
- **-i *interface***. Listen on ***interface***.
- **-n**. Do not convert addresses (such as host addresses or port numbers) to names.
- **-N**. Do not print the domain name qualification of host names.
- **-S**. Print absolute, rather than relative, TCP sequence numbers.
- **-w *file***. Write the raw packets to ***file*** rather than parsing and printing them out.

You can also specify a filter expression. The most important filter expressions are already described in the Ethereal section.

For more information, see the man page **man 1 tcpdump**.

Exercise 6-4 Use tcpdump

In this exercise, you use tcpdump to capture 10 ICMP packages.

You will find this exercise in the workbook.

(End of Exercise)

Summary

Objective	Summary
1. Use ntop to Monitor Network Traffic	ntop is a web-based traffic monitor that shows network usage, similar to what the top command does. Before you can start ntop, you have to set the administrator password using **ntop -A -u wwwrun** After entering **ntop**, you can access the ntop web interface by entering the following URL: **http://*IP-address*:3000**
2. Use Nagios to Monitor Hosts, Services, and Network	Nagios is a monitoring program for hosts, services, and network. The main configuration file of Nagios is /etc/nagios/nagios.cfg. Other configuration files are in the /etc/nagios/ directory. Most of them include command definitions for checks and notifications. The hosts are configured in the file /etc/nagios/hosts.cfg. The services are configured in the file /etc/nagios/services.cfg. Contacts are configured in the file /etc/nagios/contacts.cfg.

Objective	Summary
3. Troubleshoot the Network with Ethereal and tcpdump	Ethereal is a network protocol analyzer. The main screen of Ethereal shows three text frames: ■ **packet list** (top). Contains a table showing information about the number of packets, the time when the packet is received, sender and destination, etc. ■ **packet details** (middle). Contains the content of the packet. ■ **packet bytes** (bottom). The bytes of the packet written in hexadecimal and ASCII characters. Ethereal supports two kinds of filters: ■ Capture filters ■ Display filters tcpdump can "read" all or certain packets going over the network. tcpdump also allows you to specify a filter expression. These are equal to the Ethereal filters.

APPENDIX A Prepare for the Novell CLE 10 Practicum

In this section, you work through the following scenarios to help you prepare for the Novell CLE 10 (Certified Linux Engineer) Practicum exam:

1. Install and Configure SUSE Linux Enterprise Server 10
2. Configure a DNS Server
3. Configure a DHCP Server
4. Configure a Mail Server
5. Monitor the Network

You must complete Scenario 1. You can then select any of the remaining scenarios to complete, depending on available time.

Remember that skills from all three Novell CLE 10 courses might be necessary to fulfill the required tasks.

Scenario

Digital Airlines is planning on deploying SUSE Linux Enterprise Server 10 in its IT infrastructure. During the first phase, SUSE Linux Enterprise Server 10 will be used on the back-end systems like file, web, and network-infrastructure servers.

As the network administrator for your Digital Airlines office, you (along with management) have designed a migration plan which includes the following services to be migrated to SUSE Linux Enterprise Server 10:

- DNS services on the internal network
- DHCP server
- Mail server

The network traffic should also be monitored.

You decide to start by installing and testing these services on a computer in the test lab.

Objective 1 Install and Configure SUSE Linux Enterprise Server 10

The following are tasks that need to be performed on the test server before installing and configuring services (such as DNS):

- Install SUSE Linux Enterprise Server 10 from the SUSE Linux Enterprise Server 10 installation DVD.
- Make sure that you use a partition setup that fits your needs.
- Install only necessary applications and daemons to support a DNS server, a DHCP server, and a Mail server.
- Choose a root password.
- Select LDAP authentication.
- Create two additional normal user accounts (PSmith and JWattson).
- Update the system with YOU (from server DA1) after the installation.
- Secure the GRUB boot loader with a password.
- Configure the network connection manually (with or without YaST).
- Configure runlevel 3 as the system default.
- Synchronize the system clock with DA1 using NTP.

Read through these requirements carefully; then install and configure the server.

Objective 2 Configure a DNS Server

One milestone of Digital Airlines' move to SUSE Linux is the implementation of Linux-based DNS servers. As a first step, you decide to set up a test DNS Server in your lab.

On the SUSE Linux test server in your lab, do the following:

- Configure a master DNS server for 5 test systems (DA10-DA14).
- Test your setup (you can use the command dig).
- With another student who is working on the same scenario, configure your servers to be slave DNS servers for each other.

Objective 3 Configure a DHCP Server

Most of the workstations one your network have static IP addresses. But a small pool of notebooks is available. These notebooks should get IP addresses via DHCP.

On the SUSE Linux test server in your lab, do the following:

- Install a DHCP server.
- The IP addresses 10.0.0.180 to 10.0.0.200 are reserved for DHCP.
- The DNS and gateway information should be delivered by DHCP.

Objective 4 Configure a Mail Server

Your Digital Airlines office runs an internal mail server. This server also needs to be migrated to SUSE Linux Enterprise Server 10.

On the SUSE Linux test server in your lab, do the following:

- Email that is not for users of the digitalairlines.com domain should be forwarded to the mail server of your provider with IP 135.195.220.3.
- The host prefix is removed from the mail addresses.
- All email sent from 194.95.93.10 should be rejected.
- Email to the address info@digitalairlines.com should be sent to the user PSmith.
- The user PSmith should get a copy of all system notification sent to user root.
- Spam email should be marked with "***spam***" in the subject.
- All virus email should be sent to a user "virusadmin" and not to the mail recipient.
- All users can access their mailboxes using POP.
- Email to user PSmith should be sorted automatically into three different mailboxes:
    - Email to info@digitalairlines.com should be stored in the mailbox "info."
    - Email to root should be stored in the mailbox "root."
    - Email to psmith@digitalairlines.com should be stored in the mailbox "inbox."

Objective 5 Monitor the Network

To monitor the network services on your SUSE Linux Enterprise Server 10 use Nagios.

On the SUSE Linux test server in your lab, do the following:

- Define a host group “linux-servers”.
- Your test server is the only member of the “linux-servers” group.
- The following services should be monitored:
  - DNS
  - DHCP
  - Mail
- Configure the Nagios web server.

APPENDIX B Combine Samba and OpenLDAP

Samba is often combined with OpenLDAP because it allows you to build a backup domain controller (BDC) for Windows. Therefore the replication mechnism of OpenLDAP is used.

This appendix covers the following topics:

- Prepare the OpenLDAP Directory
- Understand the LDAP Options of Samba

Objective 1 Prepare the OpenLDAP Directory

If you want to access an LDAP directory service from Samba, you have to modify the file /etc/openldap/slapd.conf of your LDAP installation. Do the following:

1. Include the schema file of Samba3.

```
include         /etc/openldap/schema/samba3.schema
```

2. Allow Samba to access LDAP in an **access** statement.

   This can look like the following:

```
access to attrs=userpassword,sambaNTPassword,sambaLMPassword
  by self write
  by anonymous auth
  by * none
```

   In this example, everybody is allowed to use the different kinds of passwords to authenticate. They can only be changed by the password's owner.

   All attributes can be read by everybody.

3. Make sure that the options **suffix**, **rootdn**, and **rootpw** are set correctly.

4. Check the **base** option in the file /etc/ldap.conf.

5. The **index** section may look like this:

```
index   objectClass           eq
index cn                      pres,sub,eq
index sn                      pres,sub,eq
index uid                     pres,sub,eq
index displayName             pres,sub,eq
index uidNumber               eq
index gidNumber               eq
index memberUid               eq
index   sambaSID              eq
index   sambaPrimaryGroupSID  eq
index   sambaDomainName       eq
index   default               sub
```

Objective 2 Understand the LDAP Options of Samba

Samba provides some configuration options to communicate with LDAP.

The most important are the following:

- **passdb backend**. Specify in which kind of database the user and group information are stored.

  Here, you can enter one of the following:

  - **smbpasswd**
  - **tdbsam** (smbpasswd plus NT SAM)
  - **ldapsam**

  If there is a backup domain controller (BDC) in your network, select **ldapsam** here. **mysqlsam** and **xmlsam** are also possible options, but not for primary domain controllers (PDCs).

- **ldap admin dn**. Specify the LDAP distinguished name used by Samba to retrieve user information.
- **ldap delete dn**. If this option is set to **yes**, a delete operation deletes the complete LDAP entry.

  Otherwise, only the attributes specific to Samba are deleted.

- **ldap group suffix**. This suffix is used by Samba for groups, when a group is added to the LDAP directory.
- **ldap idmap suffix**. Specifies the suffix that is used when storing imap mappings.
- **ldap machine suffix**. Specifies where workstations should be added to the LDAP directory.
- **ldap passwd sync**. If this option is set to **yes**, Samba tries to synchronize the LDAP password with the NT and LANManager hashes for normal users.

If this option is set to **no**, only the NT and LANManager passwords are updated. If this option is set to **only**, only the LDAP password is changed.

- **ldap server**. Enter the FQDN of the LDAP server here.
- **ldap ssl**. If this option is set to **On**, Samba uses LDAPS to contact the LDAP server. If this option is set to **Start_tls**, extended StartTLS operations are used. **Off** disables the use of SSL.
- **ldap suffix**. Specifies where users and workstations should be added to the LDAP directory. This option can overwritten by **ldap user suffix** and **ldap machine suffix**.
- **ldap user suffix**. Specifies where users should be added to the LDAP directory.

The following options must be added to the global section of the file /etc/samba/smb.conf:

```
passdb backend = ldapsam:ldap://da51.digitalairlines.com
ldap ssl = no
ldap suffix = “dc=digitalairlines,dc=com”
ldap admin dn = “cn=administrator,dc=digitalairlines,dc=com”
```

If you want to use SSL encryption, you can set **ldap ssl** to **yes**. In this case you should change the **passdb backend** parameter to

```
passdb backend = ldapsam:ldaps://da51.digitalairlines.com
```

The replication of LDAP servers can be used to set up a Backup Domain Controller with Samba. You can find detailed information in the file Samba3-ByExample.pdf in the Samba documentation (package samba-doc).

Summary

Objective	Summary
1. Prepare the OpenLDAP Directory	To access an LDAP directory from Samba, you have to complete the following steps: ■ Include the Samba3 schema file. ■ Allow Samba to access LDAP in an **access** statement. ■ Make sure that the options **suffix**, **rootdn**, and **rootpw** are set correctly. ■ Check the **base** option in the file /etc/ldap.conf. ■ The **index** section may be changed.
2. Understand the LDAP Options of Samba	Samba provides some configuration options to communicate with LDAP. The following options must be added to the global section of the file /etc/samba/smb.conf: ■ **passdb backend** ■ **ldap ssl** ■ **ldap suffix** ■ **ldap admin dn**

APPENDIX C Retrieve Mail with Fetchmail

You can use Fetchmail to retrieve email from a server.

The package fetchmail must be installed and configured in the file ~/.fetchmailrc.

Fetchmail provides a wide range of customization options, such as

- The processing of various mailboxes
- Fetching mail for multiple users
- Integration of filter programs
- Encrypted transmission with SSL

To start Fetchmail, use the **fetchmail** command.

In addition, the progam has numerous options which can supplied when starting the program.

By default, Fetchmail can be run by any user on the system. This means that each user can create his own configuration file in his home directory.

The user root uses Fetchmail to retrieve the email centrally in regular intervals and distribute them to the individual users. This method is set in the script /etc/ppp/poll.tcpip.

The system-wide Fetchmail configuration is done in the file

/etc/fetchmailrc

For more information about Fetchmail, see the man pages for the program. Information is also available in the /usr/share/doc/packages/fetchmail/ directory.

This objective describes how to do the following:

- Configure Fetchmail
- Understand Command Line Options

Objective 1 Configure Fetchmail

A line in the ~/.fetchmailrc file has the following structure:

poll *servername* protocol *protocol* username "*name*" password "*password*"

or, in abbreviated form:

poll *servername* proto *protocol* user "*name*" pass "*password*"

This basic form can be extended by adding **is "local_name"** containing the user to which the retrieved email can be forwarded.

In the following example, this is the user geeko:

```
poll da51.digitalairlines.com proto pop3 user "geeko.chameleon" pass
"SECRET" is "geeko"
```

If several mail servers are queried, the same details can be defined once using the defaults instruction, and then they are omitted.

For example:

```
defaults proto pop3 user "geeko.chameleon"
poll da51.digitalairlines.com pass "SECRET"
poll pop3.suse.com pass "CONFIDENTIAL"
```

The basic server options are

- **proto[col]**. Type of protocol used (such as **auto**, **pop3**, and **imap**).
- **port**. Port number.
- **timeout**. Time in seconds to wait for the server replies before fetchmail closes the connection.

The basic user options are

- **user[name]**. Name of the user on the server.
- **is**. Name of the user on the client.
- **pass[word].** Password of the user on the server.
- **keep**. Messages on the server are not deleted.
- **flush**. Old messages on the server are deleted.
- **fetchall**. Fetch both new and old email messages from the server.
- **no keep**. Delete messages that have been read from the server.
- **no fetchall**. Retrieve only new messages from the server (by default).

Apart from this, a number of global settings can be made at the beginning of the Fetchmail configuration file.

The most important of these are

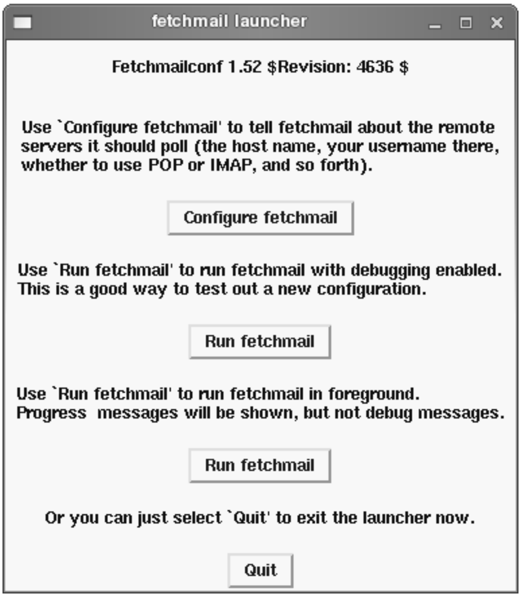
- **set daemon *seconds***. Fetchmail is started as a daemon in the background and checks at the given interval to see if email messages have arrived on the server.
- **set postmaster "*username*"**. Sets the local name of the user responsible for error messages.
- **set logfile *path***. Shows the path for the log file.

Because the file ~/.fetchmailrc contains the password for access in plain text, only the owner of the file should be granted read and write permissions (**chmod 600 .fetchmailrc**). If the permissions are too liberal, Fetchmail will terminate with a message indicating this situation.

The package **fetchmailconf** contains a graphical tool to edit the file .fetchmailrc.

The program is started by using the **fetchmailconf** command:

Figure C-1



Objective 2 Understand Command Line Options

Options that can be passed to Fetchmail in the command line override options in the file ~/.fetchmailrc with the same meaning.

The most important options are summarized in the following table:

Table C-1

Option	Description
-c, --check	Check if new messages have arrived. These are not retrieved.
-s, --silent	Suppress all status and progress messages.
-v, --verbose	All control messages are displayed on stderr.
-a, --all	Fetches all (new and old) email from the server.
-k}, --keep	Retrieved mails are not deleted on the server.
-K, --nokeep	Retrieved messages are deleted on the server.
-F, --flush	Deletes old messages from the server before the new ones are retrieved.
-p *protocol*, --protocol *protocol*	Protocol for communication with the server. *protocol* can have the values AUTO, POP2, POP3, or IMAP.
-P *number*, --port *number*	Port for the connection.
-t *seconds*, --timeout *seconds*	The time to wait for the welcome message from the server.
-u *name*, --username *name*	The name of the user.

Table C-1 *(continued)*

Option	Description
-d *seconds*, --daemon *seconds*	Fetchmail is started as a daemon and establishes a connection to the server at the interval specified.

Having only a dial-up connection to the Internet, it is useful to retrieve email from the server only when the connection is established. For this purpose, the /etc/ppp/poll.tcpip file already contains preconfigured entries.

All you need to do is to prepare a suitable Fetchmail configuration in /etc/fetchmailrc.

If the line

set syslog

exists at the beginning of /etc/fetchmailrc, the mail retrievals are logged in /var/log/mail.

Exercise C-1 Retrieve Mail Using Fetchmail

In this exercise you have to configure fetchmail to fetch the mails of user geeko from the POP postbox.

(End of Exercise)

Summary

Objective	Summary
1. Configure Fetchmail	With Fetchmail, email can be retrieved from a server. It is configured in the file **~/.fetchmailrc** To start Fetchmail, enter **fetchmail** The package **fetchmailconf** contains a graphical tool to edit the file .fetchmailrc.
2. Understand Command Line Options	Options that can be passed to Fetchmail in the command line override options in the file ~/.fetchmailrc with the same meaning.

APPENDIX D Manage Network Devices with SNMP

The Simple Network Management Protocol (SNMP) is an Internet standard for gathering statistics from and managing devices on the Internet, principally routers, but also network cards and most of the network-enabled devices.

The information about server, router, workstations, and printers is stored in the management information base (MIB). This database includes static information and dynamic information such as routing tables and memory usage.

An implemented variant of the MIB on the host that is managed by an SNMP daemon is called an agent.

An application that uses SNMP utilities to access the stored information in the MIB is called the manager.

The agent automatically collects the specified information on the hosts.

Agent and manager communicate together. The manager sends a special SNMP command to the agent and waits for the answer of the agent. For querying the agents (polling) from a management host usually UDP port 161 is used for communication, the snmp traps, the agent sends on event notifications to the specified mangagement host are sended using UDP port 162.

The most basic commands are SNMPGET and SNMPSET, to read or change information respectively. SNMPGETNEXT returns the content of the next MIB entity.

The event notifications of the agent are sent via SNMP *traps* to the manager. TRAP is the fourth important SNMP command in this section.

Another important term in the world of MIB is *community.* A community is a group of database users and their assigned permissions. You can compare the community as a string containing username and password in one and only authentication string for accessing the SNMP monitored entities.

The name of the default community is *public*. The users of public are only allowed to read from the database.

The community named *private* can read all information, but is also permitted to change information in the database (if configured).

The SNMP daemon is included in the package **net-snmp**, which is originally based on the CMU 2.1.2.1 cmu-snmp code. The net-snmp package uses various configuration files to configure its applications. These files are in /usr/share/snmp/, /etc/snmp.conf, and ~/.snmp.

Usually you would specify global setting in /usr/share/snmp and /etc/snmp.conf while you use ~/.snmp.conf for userspecific settings. Please keep in mind, that those files contain private data (passphrases and communities), so limit the read access for those files only to the appropriate user.

In this section, we cover the following topics:

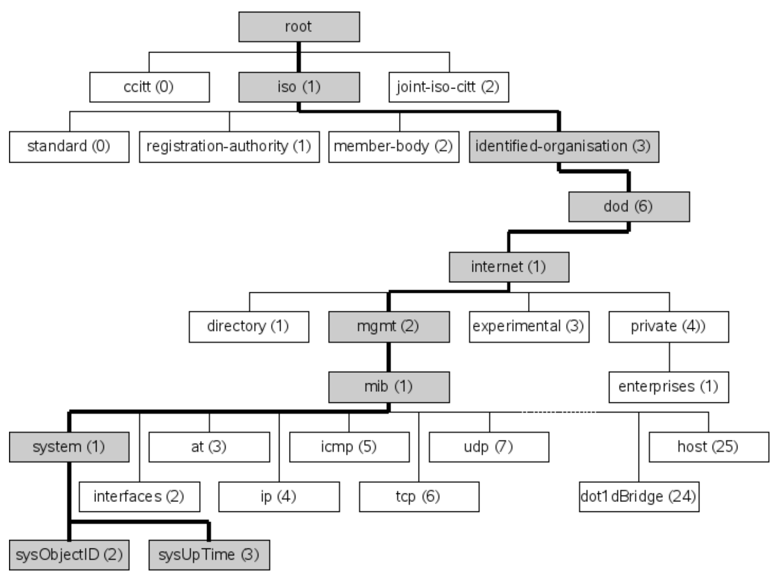
- Understand the Structure of MIB
- Configure the SNMP Daemon
- Start the SNMP Agent
- Test the SNMP Configuration

Objective 1 Understand the Structure of MIB

Each MIB can be visualized as a tree having no named root. The leaves of the tree represent the managed objects and variables. Every variable is unique and represented by its *OID* (Object Identifier). The upper layers are administered by ISO (International Organization for Standardization) and ITU (International Telecommunication Union), while lower layers are delegated to enterprises and other organizations (you can roughly compare this to the DNS, but without having a certain task or automated queries). The layer separator is the dot. Every node can be named either by its name or by the associated number so both strings iso.org.dod.internet.mgmgt.mib-2 and 1.3.6.1.2.1 specify the same node MIB-II.

The following figure shows an example of an MIB. The path to some system information is shown:

Figure D-1



The nodes at one level can be numbered. These numbers are used to identify the leaf. The following identifiers are valid for the system's object ID:

- 1.3.6.1.2.1.1.2
- 1.3.6.1.2.1.1.sysObjectID
- iso.org.dod.internet.mgmt.mib-2.1.2
- iso.org.dod.internet.mgmt.mib-2.system.sysObjectID
- iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).system(1).sysOb jectID(2)

A standarized set of common structures and identity schemas that refers to the variables of a MIB is called ~gstructure of management information~h (SMI). (RFCs 1155, 1212, 1215)

The Abstract Syntax Notation One (ASN.1) is used to describe this structure. To define a data type in ASN.1, use the following syntax:
Name* ::= *Definition

To define a managed object/variable, you have to use the SMI.1 macro **OBJECT-TYPE**. The syntax is

***Label* OBJECT-TYPE**
SYNTAX ***Data type***
MAX-ACCESS ***Permission***
STATUS ***Importance***
DESCRIPTION ***Description***

The parts of this definition are:

- **SYNTAX**. Specify the data type of the object/variable.
- **MAX-ACCESS**. Specify the access permission for the new object/variable. Possible values are:
  - **read-only**. Value is not writable.
  - **read-write**. Value is writable.
  - **write-only**. Value is not readable but writeable.

  - **not-accessible**. Value is not accessible.
- **STATUS**. Specify the importance of the object/variable. Possible values are:
  - **Mandatory**. Variable must be set.
  - **Optional**. Variable is optional.
  - **Deprecated**. Variable is replaced by another one.
  - **Obsolete**. Variable is not used anymore in this MIB.
- **DESCRIPTION**. A brief description of the object/variable.

A couple of MIBs can be found in the directory /usr/share/snmp/mibs/.

Objective 2 Configure the SNMP Daemon

The SNMP daemon is configured in the file **/etc/snmp/snmp.conf**.

After the installation, this file consists of comments and the following three lines:

```
...

syslocation Server Room
syscontact Sysadmin (root@localhost)

...

rocommunity public 127.0.0.1

...
```

A more complete example configuration file can be seen in the package documentation (/usr/share/doc/packages/net-snmp/EXAMPLE.conf).

The most important options are described in the following:

- **rouser**. Specifies a user with read permissions.
- **rwuser**. Specifies a user with write permissions.
- **rocomminity**. Specifies a community with read permissions.
- **rwcommunity**. Specifies a community with write permissions.
- **trapsink**. Alerts of the agent are sent as type of v1 notifications.
- **trap2sink**. Alerts of the agent are sent as type of v2 notifications.
- **informsink**. Alerts of the agent are sent as type of v2 inform notifications.
- **trapcommunity**. Sends an alert to the agent of this community.

- **authparenable**. If set to 1, a trap is sent when an authentication error occurs.
- **proc**. Checks for specified processes that should be running.
- **disk**. Checks the named disks mounted at the specified path for available disk space.
- **load**. Checks for unreasonable load average values.
- **file**. Checks on the size of the specified file.
- **syslocation**. The physical location of the system.
- **syscontact**. The contact information for the administrator.

This configuration file can also be created with the command **snmpconf** which leads you through setting up a configuration step by step. It creates a configuration file for the SNMP daemon in the actual directory.

```
da51:~ # snmpconf

I can create the following types of configuration files for you.
Select the file type you wish to create:
(you can create more than one as you run this program)

   1:  snmpd.conf
   2:  snmp.conf
   3:  snmptrapd.conf

Other options: quit

Select File: 1

The configuration information which can be put into snmpd.conf is divided
into sections.  Select a configuration section for snmpd.conf
that you wish to create:

   1:  Access Control Setup
   2:  Trap Destinations
   3:  Monitor Various Aspects of the Running Host
   4:  Agent Operating Mode
   5:  System Information Setup
   6:  Extending the Agent

Other options: finished

Select section: 1

...
```

snmpconf supports groups of configuration entries. The following lists the available groups you get using the option -G:

```
da51:/usr/share/doc/packages/net-snmp # snmpconf -G

Known GROUPs of tokens:

  system_setup
  basic_setup
  monitoring_services
  access_control
  trapsinks
```

To configure a group of configuration entries, enter **snmpconf -g *groupname***

For a basic SNMP setup, the command is **snmpconf -g basic_setup**

Objective 3 Start the SNMP Agent

snmpd is an SNMP agent which binds to a port and awaits requests from SNMP management software.

Upon receiving a request, it processes the request(s), collects the requested information and/or performs the requested operation(s) and returns the information to the sender.

smnpd knows a lot of options. The most important are:

- **-a**. Log the source addresses of incoming requests.
- **-A**. Append to the log file rather than truncating it.
- **-c *file***. Read ***file*** as a configuration file.
- **-C**. Do not read any configuration files except the ones optionally specified by the **-c** option.
- **-d**. Dump (in hexadecimal) the sent and received SNMP packets.
- **-D[*token*[,...]]**. Turn on debugging output only for the specified tokens. Without any tokens specified, it defaults to printing all the tokens.

To get a list of all compiled modules, run the agent with the arguments **-Dmib_init -H**.

- **-g GID**. Change to the numerical group ID GID after opening listening sockets.
- **-I [-]*modules***. Specifies which modules should (or should not) be initialized when the agent starts up. If the comma-separated list of modules is preceded with a “-”, it is the list of modules that should not be started.
- **-L[efos]**. Specify where logging output should be directed.. The possible options are:

  - **e**. Log messages to the standard error stream.
  - **f *file***. Log messages to the specified file.
  - **o**. Log messages to the standard output stream.
  - **s *facility***. Log messages via syslog, using the specified facility.
- **-m *MIBList***. Specifies a colon-separated list of MIB modules to load for this application.
- **-M *DirList***. Specifies a colon-separated list of directories to search for MIBs.
- **-p *file***. Save the process ID of the daemon in the specified files.
- **-q**. Print simpler output for easier automated parsing.
- **-r**. Do not require root access to run the daemon. This option prevents snmpd from exiting if it doesn't have permission to read something.
- **-u *UID***. Change to the user ID after opening listening sockets.
- **-V**. Symbolically dump SNMP transactions.
- **-x *address***. Listens for AgentX connections on the specified address.
- **-X**. Run as an AgentX subagent rather than as an SNMP master agent.
- **--*token*=”*value*”**. Allows you to specify any ***token*** supported in the snmpd.conf file and sets its value to ***value***.

Many utilities are included in the net-snmp package. Most of them are elementary utilities to access the MIB:

- **/usr/bin/snmpget**. Get information from the MIB.
- **/usr/bin/snmpset**. Set information in the MIB.
- **/usr/bin/snmpnetstat**. Show network status using SNMP (similar to **netstat**).

- **/usr/bin/snmptranslate**. Translates the object information values into human readable terms.
- **/usr/bin/snmptrap**. Informs the manager program about traps of the agent. The daemon version /usr/sbin/snmptrapd receives and logs SNMP trap messages.
- **/usr/bin/snmpwalk**. Incremental request of a MIB (sub-)tree.

Objective 4 Test the SNMP Configuration

For the examples in this objective the configuration file /etc/snmp/snmp.conf has to look like the following:

```
syslocation My Desk                   # Location of the system
syscontact  BOFH (root@localhost)     # responsible administrator
rocommunity public 127.0.0.1          # allow public read-only community
#                                       from loopback address
rocommunity public 147.2.90.0/23      # allow public read-only community
#                                       from 147.2.90.0/23 network
rwcommunity mysecret 127.0.0.1       # allow private read-write community
#                                       from loopback address
rwcommunity mysecret 147.2.90.0/23    # allow private read-write
#                                       community from 147.2.90.0/23
#                                       network
proc nmbd 1 1                         # check ps process so that exaclty
#                                       one instance is running
trapcommunity mysecret                # specify the community used for
#                                       snmp traps
trapsink 147.2.90.112                 # specify the trap target host
defaultMonitors yes                  # run default monitors for snmp trap
#                                       alarming
```

If you cannot remember all allowed parameters, you can run the command **snmpd -H** to view a full list of directives and their allowed values.

First make sure if the SNMP daemon is running:

```
da51:/etc/snmp # rcsnmpd status
Checking for service snmpd:                                        running
da51:/etc/snmp #
```

If it is not started yet, start it using **rsnmpd start**.

Take a look at the log file /var/log/net-snmpd.log (you may see errors).

Now you should be able to query the entries using the command **snmpget**:

```
da51:/etc/snmp # snmpget -v 1 -c public 127.0.0.1 system.sysContact.0
SNMPv2-MIB::sysContact.0 = STRING: BOFH (root@localhost)
```

The used options are:

- **-v *version***. Version number of the used protocol.
- **-c *community***. Name of the community.

An overview about the options of many SNMP commands can be found in **man 1 snmpcmd**.

You can also monitor your system using the SNMP tool set.

Let's check the networking statistics for the eth0 device:

```
da51:/etc/snmp # snmpnetstat -v 2c -c public 10.0.0.51 -Can
Active Internet (tcp) Connections (including servers)
Proto Local Address          Remote Address         (state)
tcp   *.111                   *.*                   LISTEN
tcp   *.631                   *.*                   LISTEN
tcp   *.5801                  *.*                   LISTEN
tcp   *.5901                  *.*                   LISTEN
tcp   127.0.0.1.25            *.*                   LISTEN
tcp   127.0.0.1.199           *.*                   LISTEN
tcp   127.0.0.1.427           *.*                   LISTEN
tcp   127.0.0.1.2544          *.*                   LISTEN
tcp   147.2.90.112.427        *.*                   LISTEN
Active Internet (udp) Connections
Proto Local Address
udp   *.111
udp   *.161
udp   *.631
udp   *.32772
udp   147.2.90.112.427
udp   224.0.1.22.427
udp   239.255.255.253.427
udp   255.255.255.255.427
```

You can use the **snmpdf** command to check the amount of free disk space:

```
da51:/etc/snmp # snmpdf -v 1 -c mysecret localhost
Description                 size (kB)            Used       Available Used%
Memory Buffers                2076472          128400         1948072    6%
Real Memory                   2076472          605540         1470932   29%
Swap Space                    2097136               0         2097136    0%
/                            20641788         1748652        18893136    8%
/sys                                0               0               0    0%
/sys/kernel/debug                   0               0               0    0%
/boot                         1035660           55592          980068    5%
/sys/kernel/security                  0               0               0    0%
da51:/etc/snmp #
```

You will find more information about snmpdf in **man 1 snmpdf**.

Exercise D-1 Manage Network Devices with SNMP

In this exercise you have to set up an SNMP daemon using protocol version 1.

You will find this exercise in the workbook.

(End of Exercise)

Summary

Objective	Summary
1. Understand the Structure of MIB	The Simple Network Management Protocol (SNMP) is an Internet standard for gathering statistics from and managing devices, principally routers, on the Internet. The information about servers, routers, workstations, and printers are stored in the management information base (MIB). An agent is an implemented variant of the MIB on the host that is managed by an SNMP daemon. A *manager* is an application that uses SNMP utilities to access the stored information in the MIB. The event notifications of the agent are sent via *SNMP traps* to the manager. A *community* is a group of database users and their assigned permissions. Each MIB can be visualized by a tree having no named root. The leafes of the tree represent the managed objects and variables. Every variable is unique and represented by its *OID* (Object Identifier).

Objective	Summary
2. Configure the SNMP Daemon	The Net-SNMP package uses various configuration files to configure its applications. They can be found in /usr/share/snmp/, /etc/snmp.conf, and ~/.snmp. The file /etc/snmp.conf can be created with the command **snmpconf**, which leads you through setting up a configuration step by step.

Objective	Summary
3. Start the SNMP Agent	snmpd is an SNMP agent which binds to a port and awaits requests from SNMP management software. Many utilities are included in the net-snmp package. Most of them are elementary utilities to access the MIB: ■ **/usr/bin/snmpget**. Get information from the MIB. ■ **/usr/bin/snmpset**. Set information in the MIB. ■ **/usr/bin/snmpnetstat**. Show network status using SNMP (similar to **netstat**). ■ **/usr/bin/snmptranslate**. Translates the object information values into human readable terms. ■ **/usr/bin/snmptrap**. Informs the manager program about traps of the agent. The daemon version /usr/sbin/snmptrapd receives and logs SNMP trap messages. ■ **/usr/bin/snmpwalk**. Incremental request of a MIB (sub-)tree.

Objective	Summary
4. Test the SNMP Configuration	If you cannot remember all allowed parameters, you can run the command **snmpd -H** for getting a full list of directives and their allowed values. After starting rsnmpd you can take a look at the log file /var/log/net-snmpd.log (maybe you see errors). An overview about the options of the most snmp commands can be found in **man 1 snmpcmd**. You can also monitor using the SNMP tool set.

Index

A

B

C

D

E

F

G

H

I

L

M

N

T

U

V

W

Y

Z